Kontextsensitives Routing – Architekturkonzept –

Interner Forschungsbericht
vorgelegt

von

Dipl.-Ing. Maik Debes

05. Juni 2007

Inhaltsverzeichnis

1. Aufbau und Funktionsweise

Das Gesamtkonzept der Routingarchitektur ist in Abbildung 1.1 dargestellt. Die dort dargestellten Clients stellen die Dienstnutzer dar. Die Server sind die Dienstleister. Das Netzwerk wird symbolisch durch drei Subnetze repräsentiert, die über den Kontextrouter miteinander verbunden sind. Der Begriff Server wird an dieser Stelle erweitert. Allgemein ist damit ein System gemeint, das einen Dienst anbietet. Bei den Servern können im Wesentlichen drei Typen unterschieden werden. Zum einen gibt es die herkömmlichen Server (Server 2), die keine kontextsensitiven Dienste anbieten. Des Weiteren gibt es Server (Server 5), die sowohl kontextsensitive als auch herkömmliche Dienste unterstützen und letztlich solche (Server 1, 3, 4), die ausschließlich kontextsensitive Dienste bereitstellen. Die beiden letzteren Typen werden auch als kontextsensitive Server bezeichnet. Die Unterscheidung der einzelnen Server spielt dann eine Rolle, wenn auf eine Anfrage eines Clients hin der passende Dienst gefunden und zugeordnet werden muss. Abschließend ist zu den Servern zu erwähnen, dass diese sich sich beim kontextsensitiven Router registrieren müssen. Außerdem müssen sie dort angeben, welche Kontexttypen unterstützt werden. Der Router ist erst daraufhin in der Lage, Anfragen bezüglich eines kontextsensitiven Dienstes an die entsprechenden Server weiterzuleiten.

Möchte nun ein Nutzer einen kontextsensitiven Dienst nutzen, ruft er diesen mit seinem Gerät ab. Dazu wird auf dem Endgerät eine entsprechende Anwendung gestartet, die eine Anfrage mit Angabe der zur Verfügung stehenden Kontexttypen an den Router sendet. Dieser stellt die zentrale Komponente des Architekturkonzeptes dar. Die genaue Funktionsweise wird später im Kapitel 3 detailliert beschrieben. Der Router wertet anhand des angeforderten Dienstes und der mitgesendeten Kontexttypen aus, welcher Server als Diensterbringer in Frage käme. Wurde ein passender Server gefunden (Nutzer 1 $\Rightarrow$ Server 3, Nutzer 2 $\Rightarrow$ Server 4) und dieser durch die Clients bestätigt, sendet der Router die nun folgenden Dienstanfragen direkt an den Server weiter. Er arbeitet damit im weiteren Verlauf als Proxy zwischen Client und Server. Dieses Szenario kann auch als „Proxykommunikation“ betrachtet werden.

Bei der Proxykommunikation kann sofort auf Serverausfälle reagiert werden. Ist ein Server nicht mehr erreichbar, kann der Router anhand der gespeicherten Daten einen anderen Server auswählen, der aber den gleichen Dienst anbieten muss. Diese Aus-

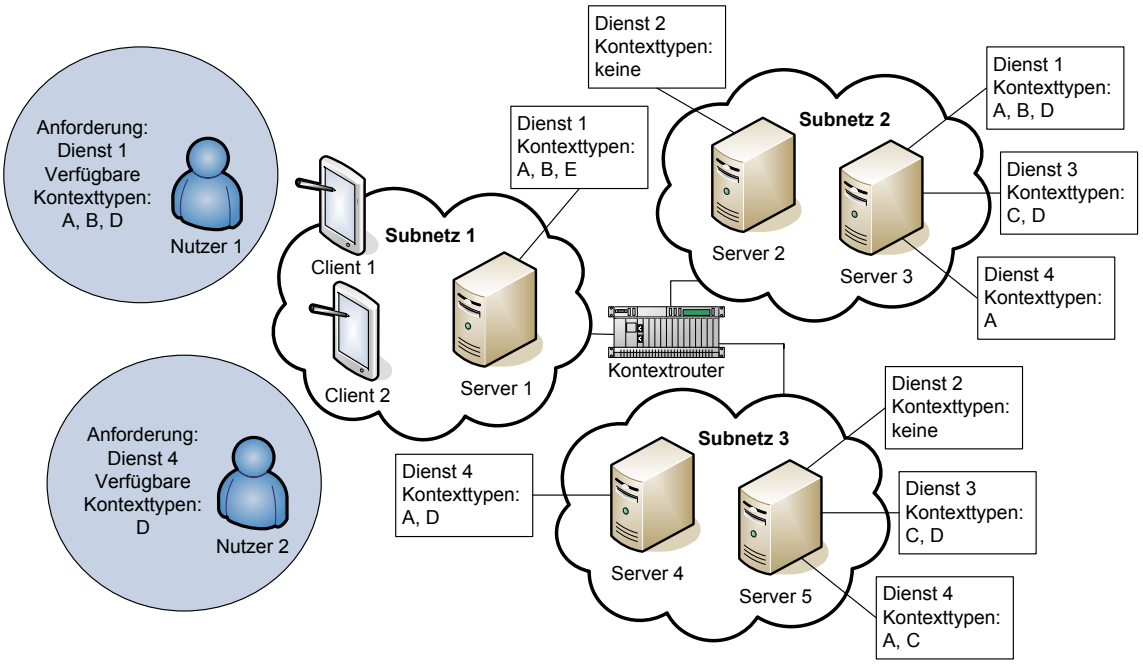


Abbildung 1.1: Die Komponenten der kontextsenstiven Routingarchitektur

wahl ist prinzipiell zu jedem Zeitpunkt möglich. Damit können auch auf einfache Weise Backuplösungen implementiert werden. Man könnte die Flexibilität des Systems so weit erhöhen, dass in bestimmten Abständen überprüft wird, ob der verwendete Server noch optimal für die aktuell zu berücksichtigenden Kontexttypen ist oder ob ein anderer Server bessere Ergebnisse liefert. Tritt dieser Fall ein, wird zu dem anderen Server gewechselt. Damit erhält der Nutzer immer den für ihn am besten angepassten Dienst. Der Server bleibt für den Nutzer zu jedem Zeitpunkt „unsichtbar“. Da der gesamte Verkehr der Kommunikation über den Router geführt wird, kann dieser auch Verwaltungs- und Monitoringfunktionen übernehmen. An dieser Stelle wären z.B. Funktionen zum Abrechnungsmanagement denkbar. Die Proxykommunikation hat aber auch Nachteile. So erhöht sich beispielsweise die Komplexität des Routers, da dieser ein Großteil der Funktionalität für die Kommunikation erbringt. Damit muss auch seine Rechenleistung entsprechend hoch sein. Da der Server „unsichtbar“ ist, hat der Nutzer nur begrenzten Einfluss auf dessen Auswahl. Dies könnte bei unseriösen Providern durchaus zu Problemen führen. Zwar wird bei den Routern mehr Leistung und damit Energie benötigt. Das stellt aber kein Problem dar, da davon auszugehen ist, dass die Kontextrouter (wie auch herkömmliche Router) nicht mobil sein werden. Die Kommunikationsabläufe selbst werden im Abschnitt 3 detailliert behandelt. Zuvor wird im folgenden Abschnitt das Routingverfahren beschrieben, was im hier dargestellten Konzept als Basis dient.

2. Routingverfahren

Um die im Abschnitt 1 beschriebenen Funktionen realisieren zu können, müssen die einzelnen Netzwerkkomponenten miteinander kommunizieren können. Die Basis dafür bildet das kontextsensitive Routing selbst. Als Basis dient hierbei das AODV (Ad hoc On Demand Distanct Vector). Dies wird kurz beschrieben und danach in Abschnitt 2.2 erklärt, wie mit Hilfe dieses Protokolls die Kontexttypen und der vom Nutzer angeforderte Dienst übertragen werden.

2.1 Funktionsweise des Basisroutingverfahrens

Um später die im Abschnitt 3 beschriebenen Kommunikationsabläufe nachvollziehen zu können, wird an dieser Stelle kurz darauf eingegangen, wie das AODV-Protokoll funktioniert. AODV ist ein tabellenbasiertes, reaktives Routingverfahren. Bei einem solchen Routingverfahren wird erst dann eine Route gesucht, wenn auch Daten zu einem bestimmten Ziel übertragen werden sollen. AODV ist im RFC 3561 [BRPD03] spezifiziert. Es ist schleifenfrei und bestimmt die Routen für eine Unicast-Kommunikation. Für die Routensuche wird als Transportprotokoll UDP verwendet. In den Netzknoten bzw. Routern, die AODV unterstützen, existieren Routingtabellen, die eine spätere bidirektionale Kommunikation ermöglichen. Man unterscheidet hierbei zwischen dem eigentlichen Routing und dem Reverse Routing. Die Tabelleneinträge für das Routing werden zur Wegewahl zum Zielknoten hin genutzt. Die Einträge für das Reverse Routing dienen dazu, von anderen Knoten erhaltene Antworten (*Route Reply*bzw. RREP an den Initiator der Wegesuche zurückzusenden. Nichtsdestotrotz können natürlich auch Daten über solche Routen gesendet werden. In der Routingtabelle werden neben der Zieladresse beispielsweise die Anzahl der Hops bis zum Ziel, das Netzwerkinterface, die Lebensdauer des Eintrags, verschieden Flags (z. B. valid, invalid, ...) usw. verwaltet. Des Weiteren wird noch eine Tabelle angelegt, die die empfangenen Anfragen (*Route Requests* bzw. RREQs) dokumentiert. Durch diese History-Funktion ist es möglich, Duplikate eines *Route Request*-Paketes zu entdecken und zu verwerfen. Letztlich wird noch eine Liste angelegt, die diejenigen Nachbarknoten enthält, die aktuell kein *Route Reply*-Paket empfangen können. Das ist notwendig, da bei mehreren von verschiedenen Nachbarknoten empfangenen identischen RREQ-Paketen, nur das erste ausgewertet wird.

Dieses RREQ könnte aber von einem Nachbarknoten initiiert worden sein, der rückwärts keine RREP-Pakete empfangen kann. Damit würde kein Weg zum Ziel gefunden werden, da alternative Routen ignoriert würden. Die in der so genannten *Black List* aufgenommenen Einträge sind nur für eine bestimmte Dauer gültig. Diese Dauer (*Timeout*) wird ebenfalls in der Liste vorgehalten.

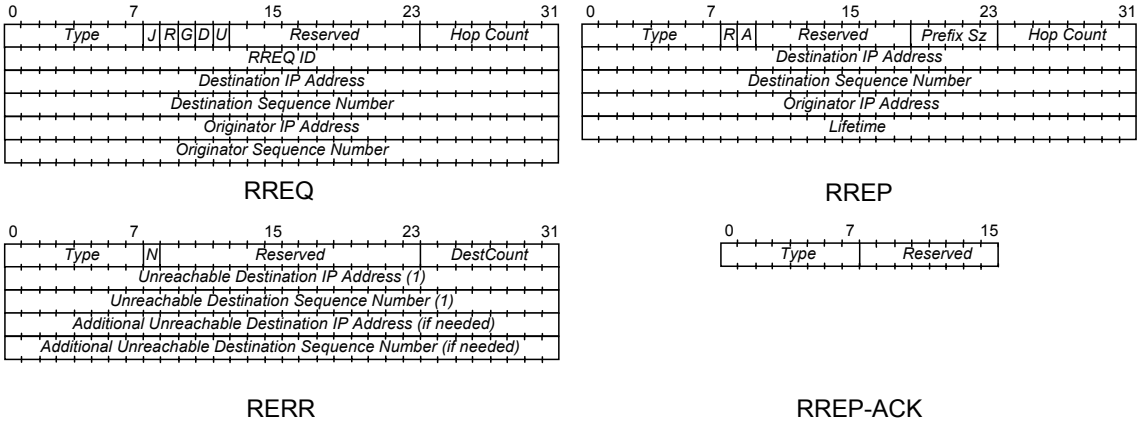


Abbildung 2.1: AODV-Paketformate

Insgesamt werden zum Finden und Managen von Routen vier Nachrichten genutzt. Der Aufbau dieser Nachrichten ist in Abbildung 2.1 dargestellt. Die Nachricht für ein RREQ wird verwendet, um einen Knoten (den Zielknoten) und damit eine Route zu diesem zu finden. Mit einem RREP erfolgt die Antwort, sobald der Zielknoten bzw. der Weg zum Zielknoten gefunden wurde. Sobald mit dem RREP eine Bestätigung für dessen Empfang gefordert wird, erfolgt dies mit einem *Route Reply Acknowledgement* (RREP ACK). Schließlich wird ein *Route Error* (RERR) gesendet, wenn bestehende Routen unterbrochen wurden. Die Bedeutung der in den Nachrichtenformaten verwendeten Felder werden im Folgenden kurz erläutert:

Type - beschreibt den Typ des Nachrichtenformats (z. B. „1“ = RREQ; „2“ = RREP; „3“ = RERR; „4“ = RREP-ACK).

J - *Join*-Flag für Multicast

R - *Repair*-Flag für Multicast

G - *Gratuitous*- Flag kennzeichnet, dass ein so genanntes Gratuitous-RREP zum im Feld „Destination IP Address“ angegebenen Knoten gesendet werden soll. Dies dient zur Einrichtung einer bidirektionalen Route, da dadurch der Zielknoten sofort die Route zum Initiatorknoten erhält.

D - *Destination Only*-Flag kennzeichnet, dass nur der Zielknoten auf dieses RREQ antworten soll.

U - wird gesetzt, wenn die Sequenznummer für den Zielknoten unbekannt ist (Unknown Sequence Number).

A - wird gesetzt, wenn die Übertragung unzuverlässig oder unidirektional ist. Der Empfänger des RREP muss daraufhin ein RREP-Ack zurück senden.

N - Dieses Flag (*No Delete*) wird gesetzt, wenn ein Knoten die Wiederherstellung eines Links initialisiert hat und somit die Route in den anderen Knoten in Richtung upstream nicht gelöscht werden soll.

Reserved - reservierter und damit ungenutzter Bereich

Hop Count - Anzahl der Hops vom Initiatorknoten bis zu dem Knoten, der die Anfrage bearbeitet.

RREQ ID - kennzeichnet zusammen mit der IP-Adresse des Initiatorknotens ein RREQ eindeutig.

Destination IP Address - ist die IP-Adresse des Zielknotens.

Destination Sequence Number - ist die Sequenznummer, die der Initiatorknoten in der Vergangenheit für eine Route in Richtung Zielknoten erhalten hat.

Originator IP Address - ist die IP-Adresse des Knotens, der das RREQ-Paket initiiert hat.

Originator Sequence Number - ist die aktuelle Sequenznummer des Knotens, der das RREQ initiiert hat.

Prefix Sz - sofern der Wert (*Prefix Size*) nicht Null ist, gibt er das Subnetz an. So gekennzeichnete Routen können dann für jeden Knoten mit der gleichen Subnetzmaske wie der eigentliche Zielknoten verwendet werden.

Lifetime - ist die Zeit in Millisekunden nach dem Empfang des RREP-Paket, für die eine Route als valid gilt.

DestCount - ist die Anzahl nicht erreichbarer Ziele und muss mindestens 1 betragen.

Unreacheable Destination IP Address (1) - ist die IP-Adresse des nicht erreichbaren Knotens. Weitere IP-Adressen können ebenfalls angegeben werden (Feld: *Additional Unreachable Destination Sequence Numbers (if needed)*).

Unreachable Destination Sequence Number (1) - ist die Sequenznummer, die in der Routingtabelle dem nicht erreichbaren Zielknoten zugeordnet ist. Sofern weitere IP-Adressen nicht erreichbarer Knoten angegeben werden, müssen natürlich auch die zugehörigen Sequenznummern angegeben werden (Feld: *Additional Unreachable Destination IP Addresses (if needed)*).

AODV zeichnet sich durch einen kleinen Overhead aus, was die Verkehrslast positiv beeinflusst. Die Funktionsweise von AODV soll anhand einer Routensuche allgemein erläutert werden. Fehler- und Ausnahmesituationen werden dabei vernachlässigt. Wie sich das Protokoll in solchen Fällen verhält, kann im RFC 3561 nachgelesen werden. In Abbildung 2.2 ist ein Beispiel für eine Routensuche dargestellt. Dort möchte Knoten A Daten an Knoten I senden. Dazu wird ein RREQ an alle Nachbarn gesendet. Direkt erreichbare Nachbarn sind in der Abbildung mit einer Linie verbunden. Diese leiten die Anfrage wiederum an ihre Nachbarn weiter, sofern der Zielknoten nicht bekannt ist oder das *D*-Flag gesetzt ist. Duplikate werden dabei erkannt und

gelöscht. Jeder Knoten der ein RREQ erhält, speichert die Routeninformation zum rückwärtigen Erreichen des Initiators der Anfrage ab. Wurde der Zielknoten erreicht (Bild d in Abbildung 2.2), sendet dieser ein RREP an den Initiator (Knoten A) zurück. Das RREP findet den Weg zur Quelle über die Reverse-Route-Einträge der Knoten. Gleichzeitig aktualisiert jeder Knoten, der das RREP weiter leitet, seine Routingtabelle und kennt damit auch den Weg zum Knoten I.

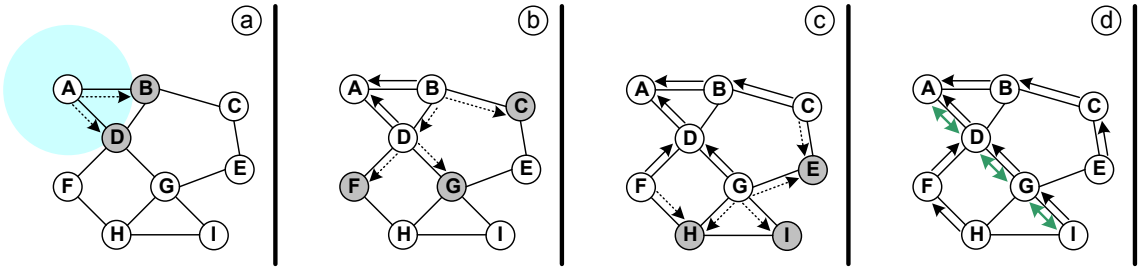


Abbildung 2.2: AODV-Routingalgorithmus [Pasc05]

Ergänzend sei noch erwähnt, dass der RFC 3561 AODV basierend auf IPv4 beschreibt. Mittlerweile befindet sich auch das „*Ad hoc On-Demand Distance Vector (AODV) Routing for IP version 6*“ [PeRD00] in der Standardisierungsphase. Prinzipiell arbeitet dort das Routingverfahren genauso wie unter IPv4. Die Paketformate wurden dahingehend erweitert, dass nun 128-Bit-lange Quell- und Ziel-IP-Adressen (*Source IP Address, Destination IP Address*) verwendet werden. Beim RREQ wird des Weiteren auf die Flags *D* und *U* verzichtet. Damit erweitert sich der reservierte Bereich um 2 Bit. Die *RREQ ID* wird durch die gleichlange *Flooded Packet ID* ersetzt. Da IPv6 keine Broadcast-Adressen kennt, wird zur Routensuche mit Multicast gearbeitet. Hierzu ist in der Spezifikation eine Multicastadresse definiert, die im aktuellen Subnetz wie ein Broadcast funktioniert, sodass alle Knoten des Subnetzes erreicht werden können. Nähere Details zur Konfiguration des AODV unter IPv6 können in oben genannter Quelle nachgelesen werden.

2.2 Erweiterung des Basisroutingverfahrens

Das in dieser Arbeit vorgestellte Konzept sieht nur die Übertragung von Kontexttypen mit Hilfe des AODV vor. Die Übertragung der zu den Kontexttypen zugehörigen Werte wird als wenig sinnvoll erachtet, da dies Aufgabe der späteren Kommunikation ist. Die Anfrage mittels RREQ-Paket dient lediglich dazu einen passenden Server zu finden, der den angeforderten Dienst erbringen bzw. die angegebenen Kontexttypen verarbeiten kann. Die Kontextwerte werden also später zwischen Client und kontextsensitiven Server direkt ausgetauscht, sodass der Wegewahlprozess diesbezüglich entlastet wird.

AODV lässt sich dahingehend sehr gut für zusätzliche Informationen modifizieren, da es in verschiedenen Nachrichtenformaten Bits vorhält, die unbenutzt sind. In Abbildung 2.1 sind das die Felder „Reserved“. Des Weiteren bietet das Protokoll nach RFC 3561 die Möglichkeit die RREQ- und RREP-Pakete zu erweitern. Somit können bis zu 255 Byte an zusätzlichen Daten übertragen werden. Dazu wird das Paketformat um die in Abbildung 2.3 dargestellten Anteile erweitert. Mit dem Feld Type kann eine Kategorisierung der angehängten Daten erfolgen. Der Wert 1 ist hierbei schon für das *Hello Interval Extension*-Format vergeben. Das *Length*-Feld

gibt die Anzahl der zu übertragenden Daten in Bytes an. Das Feld *type-specific data* enthält schließlich die eigentlichen Daten.

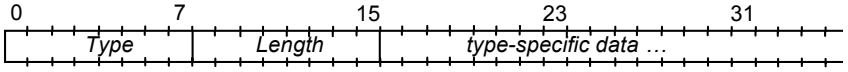


Abbildung 2.3: Erweiterung für RREQ- und RREP-Pakete

Reservierte Bereiche im Nachrichtenformat gibt es für RREQ- (11 Bit bei IPv4, 13 Bit bei IPv6), RREP- (9 Bit bei IPv4, 7 Bit bei Ipv6) und RERR-Pakete (15 Bit für IPv4 und IPv6). Da der Teilnehmer bei einer Dienstsuche als erstes eine entsprechende Anfrage an das Netz sendet, sind an dieser Stelle auch die Kontexttypen und der gewünschte Dienst mit zu übertragen. Ein Netzknoten, der diese Anfrage dann auswerten kann (also der Kontextrouter), liest die gesuchten Kontexttypen aus und verarbeitet diese entsprechend. Aus dieser Funktionsweise heraus ergibt sich, dass lediglich das RREQ-Paket zur Übermittlung der zusätzlichen Informationen in Frage kommt. Mit diesem Paket stünden elf freie Bits zur Verfügung. D. h., es könnten theoretisch bis $2^{11} - 1 = 2047$ verschiedene Kontexttypen unterstützt werden. Aktuelle Entwicklungen und Forschungsarbeiten deuten aber darauf hin, dass kontextsensitive Anwendungen mehrere Kontexttypen gleichzeitig berücksichtigen und benötigen. Um beliebige Kombinationen von diesen Typen mittels RREQ übertragen zu können, müsste pro Kontexttyp ein Bit reserviert werden. Das bedeutet jedoch, dass nur 11 verschiedene Kontexttypen verwendet werden können. Diese Anzahl reicht aber bei weitem nicht aus. Hinzu kommt, dass mit der Weiterentwicklung der Technik eine immer genauer werdende Beschreibung des Kontextes möglich wird. Kontexttypen, die heute vielleicht messtechnisch noch nicht erfassbar sind (z. B der mentale Zustand eines Individiums), können dann erfasst und ausgewertet werden. Diese Eigenschaft wird die Zahl der verwertbaren Kontexttypen auch zukünftig noch ansteigen lassen.

Eine bessere Alternative gegenüber der Nutzung der reservierten Bits bietet die bereits erwähnte Erweiterungsmöglichkeit von RREQ- und RREP-Pakete. Ein solcher Vorschlag wird auch in [KoPe02] unterbreitet. Allerdings sind dort in den Paketerweiterungen keine Möglichkeiten vorgesehen um Kontexttypen zu übertragen, womit eine Nutzung innerhalb der kontextsensitiven Routingarchitektur ausscheidet. Mit den in den Erweiterungen der RREQ- und RREP-Paketen zur Verfügung stehenden 255 Bytes können theoretisch bis zu $255 * 8 - 1 = 2039$ verschiedene Kontexttypen gleichzeitig übertragen werden. Zusätzlich müssen auch noch der gewünschte Dienst und eventuell zusätzliche Informationen berücksichtigt werden, sodass sich diese Zahl entsprechend verringert. Trotzdem bietet diese Möglichkeit auch in absehbarer Zukunft genügend Reserven. Zu beachten ist jedoch, dass sich die Größe der RREQ-/RREP-Pakete bei Verwendung aller Bytes um etwa 11 mal vergrößert. Eine Effizienzsteigerung diesbezüglich könnte dann durch die Nutzung des Längenfelds erfolgen, weil damit die Größe des Datenfelds an die aktuell zu übertragenden Kontexttypen angepasst werden kann. Neben diesen Feststellungen ist zu diskutieren, inwiefern es sinnvoll ist, die Daten strukturiert zu übertragen. Vorschläge dazu werden in Abschnitt 3.2.4 unterbreitet.

Damit das kontextsensitive Routing mit Hilfe von erweiterten RREQ auch tatsächlich funktioniert, ist es notwendig, dass im Paketkopf das *D*-Flag per Default gesetzt

wird. Der Grund liegt in der Arbeitsweise des AODV. Empfängt ein Zwischenknoten[1] ein RREQ, so schaut dieser in seiner Routingtabelle nach, ob ihm die Route zum Ziel schon bekannt ist. Ist dies nicht der Fall, erfolgt ein Broadcast an alle benachbarten Knoten. Ist diesem Zwischenknoten jedoch schon eine Route zum Ziel bekannt, so antwortet er sofort mit einem RREP. Mit dem Setzen des *D*-Flags wird nun sicher gestellt, dass das RREQ tatsächlich zum Ziel weitergeleitet wird. Dies ist natürlich auch in den Festnetzen interessant, da sich hier die Routen kaum ändern. Das wirkt sich auch auf die Kompatibilität bei Verwendung des IPv6 aus, da in dem gegenwärtigen *Internet Draft* [PeRD00] kein *D*-Flag vorgesehen ist. Vielmehr werden dort zusätzlich modifizierte RREQ- und RREP-Paketformate angeboten, mit denen 255 Bytes Nutzdaten direkt zum Ziel gesendet werden können. Bezeichnet werden diese als Optionen/Alternativen. Eine Anfrage erfolgt dann mit dem *Flood Data Option*-Format und eine Antwort mit dem *Flood Data Reply Option*-Format. Bei größeren Abständen zwischen zwei Dienstanfragen und Netzen mit dynamischer Topologie hat die Problematik der Weiterleitung des RREQ-Paketes fast keine Auswirkung. Für eine heterogene Netzumgebung, die auch stationäre Topologieanteile enthält, ist sie jedoch um so mehr von Bedeutung.

Werden die Erweiterungen der RREQ-/RREP-Pakete, wie in diesem Abschnitt beschrieben, genutzt, hat das keine Auswirkungen auf solche Router, die keine Kontextdaten auswerten können. Die Erweiterung wird im einfachsten Fall ignoriert und das Paket als solches weiter geleitet. Ein konkretes Beispiel für die erweiterten RREQ-/RREP-Pakete erfolgt später in Abschnitt 3.4, da erst im folgenden Kapitel darüber diskutiert wird, welche Funktionen von den einzelnen Netzknoten innerhalb der Routingarchitektur zu erbringen sind, und damit erst feststeht, welche zusätzlichen Informationen mit Hilfe der Pakete übertragen werden müssen.

[1] Als Zwischenknoten ist einer der routenden Knoten zwischen Sender und Empfänger gemeint.

3. Kommunikationsprozesse

Dieses Kapitel geht detailliert auf die Kommunikationsabläufe zwischen den kontextsensitiven Komponenten ein. Es wird dabei beschrieben, welche Aufgaben die kontextsensitiven Server, Kontextrouter und Clients übernehmen und welche Voraussetzungen sie erfüllen müssen. Grundlage dazu bildet im folgenden Kapitel die Diskussion darüber, wie die Adressierung im System erfolgen soll.

3.1 Adressierung

Grundsätzlich wird in einem IP-Netz gearbeitet. D. h., es werden IP-Adressen verwendet. Damit die Komponenten aber auch miteinander kommunizieren können, müssen diese die Adressen der jeweiligen Kommunikationspartner kennen. Die zentrale Komponente bildet hier der Kontextrouter. Um mit diesem kommunizieren zu können, müssen entweder die Server und Clients die IP-Adresse des Routers besitzen oder aber der Router besitzt die IP-Adressen der betreffenden Hosts. Aus verkehrstheoretischen und funktionalen Erwägungen heraus ist es sinnvoller, dass die Routeradresse den kontextsensitiven Servern und den Clients bereitgestellt wird. Zum einen fragen die Clients bei Bedarf einen kontextsensitiven Dienst an, deshalb benötigen sie die Router-IP-Adresse. Zum anderen müssen sich die Server beim Kontextrouter registrieren, wofür sie ebenfalls die Router-IP-Adresse benötigen. Die Initiierung der Kommunikation geht also immer von den Clients und Servern aus. Des Weiteren können sich die kontextsensitiven Server in anderen Subnetzen befinden. Für den Kontextrouter wären sie dann nicht „sichtbar“. Verschiedene Kommunikationsmöglichkeiten bezüglich der Adressierung wurden in [Schm05] diskutiert. Wird dabei berücksichtigt, dass für eine spätere Dienstanfrage an den Router ein erweitertes AODV-RREQ verwendet wird, ergeben sich die im Folgenden beschriebenen Kommunikationsmöglichkeiten. Dabei wird davon ausgegangen, dass die jeweils genannte Kommunikationsart vom Client bzw. Server lediglich dazu verwendet wird, um den Kontextrouter zu erreichen. Deshalb wird unter den einzelnen Punkten diskutiert, wie dieser Router eigentlich gefunden bzw. wie dessen Adresse im Netz verbreitet werden kann.

Unicast: Hierbei handelt es sich um eine Punkt-zu-Punkt-Kommunikation. Das bedeutet für den Client bzw. Server, dass er bei einer Kommunikation mit dem Kon-

textrouter dessen IP-Adresse kennen muss. Um diese Adresse im Netzwerk bekannt zu machen, stehen folgende Möglichkeiten zur Verfügung:

1. Es werden Adressen festgelegt, die für jedes Netz gelten. Diese so genannten Well-Known-Adressen sind dann reserviert und dürfen in jedem Netz nur für die Kontextrouter zur Verfügung stehen. Mit diesen Adressen werden die Router adressiert. Gleichzeitig sind sie jedem Nutzer im Netz bekannt.

2. Der Nutzer muss sich beim Systemadministrator erkundigen, welche Kontextrouter für für ihn in Frage kommen. Perspektivisch könnte dies für Netzwerke, die mit dem Dynamic Host Control Protocol (DHCP) arbeiten, automatisiert werden. Dazu müsste aber das DHCP extra um diese Möglichkeit erweitert werden.

3. Der Router sendet periodisch so genannte Advertisement-Pakete in das Netzwerk. Dadurch ist jedem Netzknoten bekannt, wie die Adresse des Routers lautet. Das Advertisement-Paket müsste per Broadcast bzw. Multicast versendet werden. Das wäre nur innerhalb des Subnetzes möglich. Um in andere Subnetze zu gelangen, müssten Relay-Agenten oder Proxys genutzt werden, die das Advertisement des Routers weiterleiten. Alternativ können die Knoten so genannte Solicitation-Pakete versenden, um die Adressen der Kontextrouter zu erhalten.

Anycast: Dieser Ansatz könnte zum Auffinden der Kontextrouter verwendet werden. Danach würde ein AODV-RREQ an alle Netzknoten (zumindest des Subnetzes) gesendet werden. Der erste Kontextrouter würde auf diese Anfrage antworten. Allerdings würde die Adressierung mit Hilfe von Anycasts voraussetzen, dass alle Kontextrouter die gleiche IP-Adresse besitzen. Des Weiteren wäre dadurch die dedizierte Auswahl eines alternativen Kontextrouters nur schwer möglich.

Multicast: Bei der Nutzung von Multicasts würden die Kontextrouter eine spezielle Multicastadresse erhalten. Hierbei handelt es sich bei IPv4 um eine Adresse aus dem Klasse-D-Netz (224.0.0.0 - 239.255.255.255). Bei IPv6 sind alle Adressen reserviert, die mit ff00::/8 beginnen. Eine Anfrage durch den Client erfolgt dann über eine solche Adresse. Alle zur Multicastgruppe gehörenden Knoten – das sind hier die Kontextrouter – antworten. Der Nutzer kann dann einen davon auswählen.

Concast: Beim Concast gibt es mehrere Sender und einen Empfänger. Im hier betrachteten Szenario tritt dieser Fall ein, wenn mehrere Nutzer parallel eine Anfrage an einen Kontextrouter senden würden. Damit ist dies ein Spezielfall des Unicast und es gelten die dort gemachten Aussagen.

Broadcast: Bei einem Broadcast sendet ein Knoten an alle anderen Knoten des Netzes. Dadurch würde bei dem hier betrachteten Konzept das gesamte Netz mit der Anfrage nach einem Kontextrouter „geflutet“. Jeder Knoten erhält die Anfrage und die Kontextrouter würden entsprechend darauf antworten. Mit Hilfe dieser Antworten kann dann der Nutzer einen Kontextrouter auswählen.

Beacons: Die so genannten Leuchtfeuer (Beacons) sind hier nur ergänzend aufgenommen. Ein Netzknoten sendet hierbei periodisch Informationen aus. Das geschieht mit Broadcasts. Zum Senden einer Dienstanfrage an einen Kontextrouter ist diese

Methode ungeeignet, da dadurch die Verkehrslast, sofern sich viele Clients und Server im Netz befinden, sehr stark steigen würde. Für den Router selber stellt dies jedoch eine Alternative dar, um anderen Netzknoten seine IP-Adresse mitzuteilen (siehe Punkt 3 bei Unicast).

Kommunikation	Adressbezug	Initiator	Adresse des Kontextrouters	Weiterleitung	AODV
Uni-/Concast	manuell	–	well-known; individuell	ja	ja
	Advertisement	Router	individuell	nein	ja
	Solicitation	Clients; Server	individuell	nein	ja
Anycast	manuell	–	well-known; individuell	ja	ja
Multicast	manuell	–	well-known	nein	ja
Broadcast	manuell	–	well-known	nein	nein

Tabelle 3.1: Gegenüberstellung der Adressierungsmöglichkeiten

Die vorgestellten Adressierungsmöglichkeiten besitzen verschiedene Eigenschaften, die in Tabelle 3.1 gegenübergestellt sind. Beacons werden dabei aus den oben genannten Gründen nicht mit betrachtet. Zu jeder Kommunikationsart wird aufgeführt, wie die Clients und Server die IP-Adresse des Kontextrouters erhalten. Dafür gibt es drei Möglichkeiten. Die IP-Adresse wird manuell vom Nutzer oder einem Administrator übergeben, der Router macht seine Adresse per Advertisement bekannt oder die Knoten suchen mit Hilfe von Solicitations nach Kontextroutern. In weiteren Spalten wird angegeben, wer die Adresssuche initiiert und ob für den Kontextrouter eine well-known- oder eine individuell festgelegte IP-Adresse verwendet werden kann. Zum Schluss wird für die jeweiligen Kommunikations- und Adressvarianten noch angegeben, ob diese eine Kommunikation über Subnetzgrenzen hinweg unterstützen und ob sie mit dem AODV-Protokoll realisiert werden können.

Die einzelnen Adressierungsmöglichkeiten werden im Folgenden weiter untersucht, um eine für das kontextsensitive Routing geeignete Methode zu finden. Betrachtet man Multicasts und Broadcasts, dann haben diese den Vorteil, dass bei der Adressanfrage durch einen Client ggf. mehrere Kontextrouter antworten. Damit kann sich der Client bzw. Nutzer den „besten“ Router aussuchen. Es ist jedoch zu beachten, dass ein Netzbetreiber wohl kaum mehr als einen Kontextrouter innerhalb eines Subnetzes aufstellen wird. Des Weiteren unterstützt IPv6 keine Broadcasts. Ebenso würden Anycasts aus ihrer Funktionsweise heraus, nur über die Subnetzgrenzen hinweg Sinn machen. Zwei Kontextrouter mit gleichen IP-Adresse können nämlich nur in örtlich getrennten Netzen gleichzeitig betrieben werden, d. h. sie müssen über mindestens einen weiteren Router voneinander getrennt sein. Zusätzlich gilt für alle hier vorgestellten Adressierungsvarianten, dass ein Überschreiten der Subnetzgrenze durch ein AODV-Paket nur durch den Einsatz spezieller Relay-Agenten bzw. Proxys erfolgen kann. Damit würde jeder Knoten und jeder Teilnehmer die Möglichkeit haben Broadcast-/Multicastanfragen über Subnetzgrenzen hinweg durchzuführen. Das würde dem Sicherheitskonzept im RFC 2644 [Seni99] widersprechen. Deshalb werden solche Ansätze bei der Konzeptionierung der Routingarchitektur verworfen. Damit

können Anycast, Multicast und Broadcasts nicht zum Finden eines Kontextrouters verwendet werden. Wie bereits erwähnt, scheiden Beacons aus funktionalen Gründen ebenfalls aus. Übrig bleiben die Unicast-Varianten, da dort die Adressen der Kontextrouter nicht über eine AODV-Anfrage ermittelt werden müssen.

Vergleicht man nun die drei Unicast-Varianten, kommt man zu folgenden Ergebnissen. Well-Known-Adressen sind jedem Nutzer im Netz bekannt. D. h., diese müssten dann auch global in den verschiedenen Netzen und Subnetzen an Kontextrouter vergeben werden können. Für IPv4 ist dies aufgrund der Strukturierung der Adressbereiche nur schwer umsetzbar. Bei IPv6 ist es ähnlich, da es auch hier unterschiedliche Subnetze gibt. Eine Lösung wäre, einen endlichen Pool von IP-Adressen aus verschiedenen Netzwerken bzw. Subnetzen bereitzustellen, um dann die einzelnen Netze bedienen zu können. Diese Lösung ist jedoch sehr unflexibel, da die Anzahl der Kontextrouter begrenzt und auf bestimmte Netze reduziert wäre. Auch die zweite Unicast-Variante gestaltet sich als unflexibel, da die Adresse des Routers vom Administrator bezogen werden müsste. Die hier vorgeschlagene Realisierung per DHCP erscheint als sehr vielversprechend, hat aber zwei entscheidende Nachteile. Zum einen besitzt DHCP keine vollständige Netzdurchdringung. D. h., viele Netze arbeiten mit statischen Konfigurationen. Zum anderen müsste die Spezifikation des DHCP erweitert werden, damit es die Clients mit zusätzlichen Informationen über Kontextrouter versorgen kann. Eine diesbezügliche Änderung der Spezifikation wird mit Sicherheit aber erst dann erfolgen, wenn sich das kontextsensitive Routing in der Praxis bewährt hat. Deshalb wird in dieser Arbeit auf die dritte Variante zurückgegriffen, bei der ein Kontextrouter die Information über seine Anwesenheit und IP-Adresse selbst verbreitet. Dazu müssen keine Protokolle oberhalb der Vermittlungsschicht (außer AODV) modifiziert werden und die Variante wäre in allen IP-Netzen anwendbar. Bleibt noch die Problematik der Subnetzgrenzen. Dabei ist jedoch zu berücksichtigen, dass Router eigentlich als Verbindungsglied verschiedener Netze oder Subnetze dienen. Broad-/Multicasts, die von einem solchen Router aus gesendet werden, können in alle angeschlossenen Subnetze weitergeleitet werden. Zwar werden dann auch die Advertisements des Kontextrouters, die als Broad-/Multicasts auf IP-Schicht gesendet werden, nicht einfach über die angeschlossenen Subnetze hinaus weitergeleitet, dennoch wird dadurch der durchdringbare Netzdurchmesser merklich vergrößert. Sollte eine noch größere Reichweite erforderlich sein, wäre zudem der Einsatz von Relay-Agenten oder Proxys, die dann nur die Advertisements der Kontextrouter weiterleiten müssten, effektiver als eine diesbezügliche Realisierung für Clients und Server. Außerdem könnte eine solche Weiterleitung besser in der Praxis umgesetzt werden. Diese Behauptung ergibt sich aus der Tatsache heraus, dass ein Provider eine Weiterleitung von Broad-/Multicasts seiner eigenen Geräte oder Geräte anderer vertrauenswürdiger Betreiber eher unterstützt und sei es nur in weitere Teilnetze seines Einzugsgebietes. Bei einer Weiterleitung von AODV-Broad-/Multicasts zum Finden eines Kontextrouters müssten alle Clients und Server dieses Recht erhalten.

Letztlich könnte man durch Verknüpfung der Unicast-Varianten 2 und 3, wie dies in Abbildung 3.1 dargestellt ist, und mit Rücksicht auf die örtliche Abhängigkeit der Anfragen nach kontextsensitiven Diensten auch ganz auf eine Weiterleitung von Broad-/Multicast über Netzgrenzen hinweg verzichten. Hierbei versendet der Router, wie im vorangegangenen Absatz beschrieben, Advertisements in die angeschlossenen Subnetze. Es wird davon ausgegangen, dass sich die Clients und Server in einem der IP-Subnetze befinden. Server, die sich außerhalb dieses Subnetzes befinden, müs-

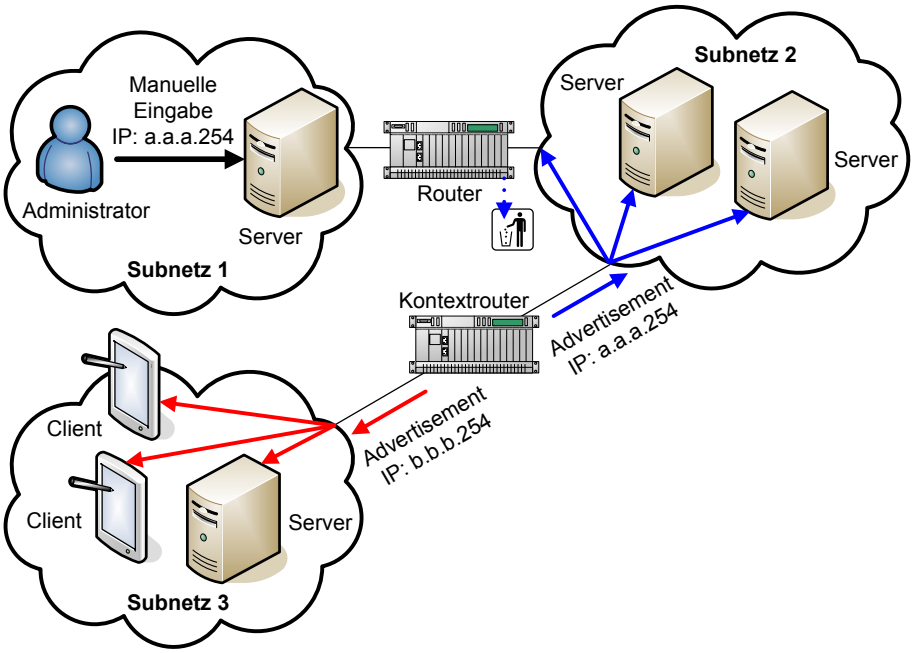


Abbildung 3.1: Die Verbreitung der Adresse des Kontextrouters

sen die IP-Adresse des Kontextrouters anderweitig beziehen (z.B. durch manuelle Konfiguration). Auch in dieser Variante sollte es jedoch ein Provider vorziehen, die Advertisements eines im eigenen Netz betriebenen Kontextrouters in die eigenen Subnetze weiterzuleiten. Diese Kombination aus den beiden Unicast-Varianten stellt gegenwärtig das beste Lösungskonzept dar. Sie wird auch, wie in Tabelle 3.1 zu sehen ist, von AODV unterstützt. Deshalb soll diese Lösung für das kontextsensitive Routing verwendet werden. Damit ergeben sich für die einzelnen Komponenten der Routingarchitektur die in den folgenden Abschnitten beschriebenen Kommunikationsprozesse.

3.2 Kontextrouter

Der Kontextrouter stellt das zentrale Element in der Routingarchitektur dar. Er registriert die kontextsensitiven Server, beantwortet Anfragen von Clients und verwaltet die Kommunikation zwischen Client und Server. Um diese Aufgaben erfüllen zu können, muss der Kontextrouter, die in den folgenden Abschnitten beschriebenen grundlegenden Funktionalitäten erfüllen.

3.2.1 Advertisements

Damit ein Netzknoten die Funktionen des Kontextrouters in Anspruch nehmen kann, muss ihm dessen IP-Adresse bekannt sein. Dazu wird sie aus den im Abschnitt 3.1 dargelegten Gründen mit Advertisement-Paketen im Netz veröffentlicht. Eine solche Möglichkeit wird bereits bei *Mobile IP* [Perk02] angewendet. Dort wird diese Funktion über eine Erweiterung des ICMP (Internet Control Message Protocol), die so genannte *ICMP Router Discovery Message* realisiert. Der zugehörige RFC 1256 [Deer91] beschreibt diese Erweiterung. Von Vorteil ist hierbei, dass ICMP sowohl für IPv4 als auch für IPv6 vorgesehen ist. In Abbildung 3.2 ist ein ICMP-Advertisement-Paket für IPv4 dargestellt. Die ersten 4 Byte entsprechen denen eines herkömmlichen ICMP-Paketes. Im Einzelnen haben die Felder folgende Bedeutung:

Type - ist der Typ der Nachricht (Advertisement = 9)

Code - Hier steht eigentlich detaillierter der Grund, warum eine ICMP-Nachricht versendet wurde. Für das Advertisement wird per Default die 0 gesetzt.

Checksum - Prüfsumme

Num Addrs - Anzahl der Router, die mit dieser Nachricht ihre Adresse veröffentlichen.

Addr Entry Size - gibt die Anzahl von 32-Bit-Wörtern wider, aus denen sich die Informationen über die jeweilige Routeradresse zusammensetzen (= Gesamtzahl der Bits aus *Router Address[i]* und *Preference Level[i]* ÷ 32)

Lifetime - ist die Dauer in Sekunden, in der Adressen der Router als gültig betrachtet werden.

Router Address[i] - sind die gesendeten Routeradressen von dem Interface, von dem diese Nachricht stammt. (i=1... *Num Addrs*)

Preference Level[i] - ist ein Wert für die Priorität der jeweiligen Adresse.

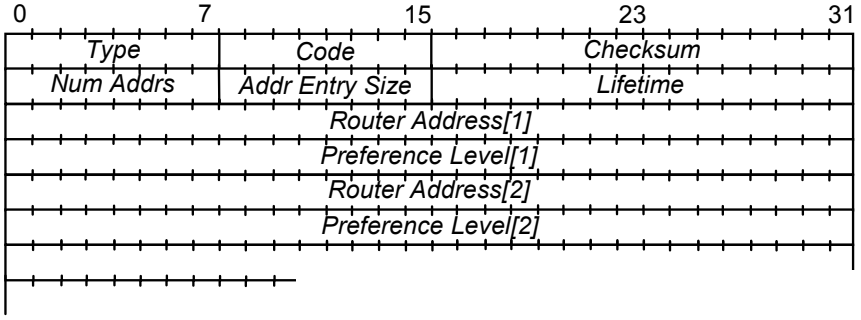


Abbildung 3.2: ICMP-Advertisment-Paket

Das Versenden eines Advertisement-Paketes reicht noch nicht aus, um den Netzknoten mitzuteilen, dass sich hinter dem Sender ein Kontextrouter verbirgt. Daraus ist lediglich ersichtlich, dass sich ein Router im Netzwerk befindet. Zur speziellen Kennzeichnung eines Kontextrouters kann beispielsweise das Feld *Code* verwendet werden. Zu beachten ist an dieser Stelle, dass nach RFC 1256 alle Hosts solche Advertisements verwerfen. Allerdings muss auf den Clients und Servern, die kontextsensitives Routing unterstützen, spezielle Software implementiert werden, die die erforderlichen Funktionalitäten ermöglichen. Dort kann auch das Verhalten des Hosts dahingehend verändert, dass Advertisement-Pakete mit einem *Code*-Feld ungleich Null nicht verworfen, sondern entsprechend ausgewertet werden. Entspricht der Wert des Feldes dem, der für einen Kontextrouter festgelegt wurde, wird die mitgelieferte IP-Adresse akzeptiert und übernommen. Somit können alle Netznoten, die kontextsensitives Routing unterstützen, diese Advertisement-Pakete auswerten. Für alle anderen Knoten sind diese Pakete transparent und werden verworfen. Damit bleibt die Kompatibilität erhalten. Ein ähnlicher Weg ist für *Mobile IP* in [Schi03] beschrieben. Dort wird neben der Mobilitätserweiterung des Advertisementpaketes das Verhalten der Netzknoten beim Empfang eines solchen Paketes mit Hilfe verschiedener Codes (0 und 16) gesteuert.

Advertisement-Pakete werden nur in dem Subnetz versendet, in dem sich der Initiator der Sendung befindet. Dazu wird entweder ein *Directed* Broadcast oder das entsprechende Multicast verwendet. Nach RFC 1256 wird dabei im IP-Paket die *Time To Live* (TTL) auf 1 gesetzt, was zwangsläufig zu Problemen bei Ad-hoc-Netzen führen muss, da hier jeder Knoten einen Router repräsentiert. Dagegen lässt AODV jedoch Multicasts explizit zu, was sowohl in [RoPe99] und [RoPe01] beschrieben und simuliert wurde. Damit ist das beschriebene Verfahren der Adressierung mittels ICMP auch in Ad-hoc-Netzen anwendbar. Beim Senden der Advertisement-Pakete wird lediglich die IP-Adresse des Kontextrouters gesendet. In späteren Erweiterungen könnten beispielsweise auch die IP-Adressen alternativer Kontextrouter übertragen werden.

Die Advertisement-Pakete müssen in bestimmten zeitlichen Abständen verbreitet werden. Nach der Spezifikation RFC 1256 muss dazu ein Wert zwischen 4 und 1800 Sekunden verwendet werden. Der Defaultwert beträgt 600 Sekunden. Natürlich besteht immer die Wahrscheinlichkeit, dass ein Knoten innerhalb dieses Zeitraums einen Kontextrouter benötigt. So ein Fall liegt zum Beispiel vor, wenn ein Rechner neu gestartet wurde. Er müsste jetzt im schlechtesten Fall die gesamte Intervallzeit auf eine Adressinformation warten. Bei Verwendung des Defaultwertes sind das 10 Minuten. Netzknoten, denen diese Zeit nicht zur Verfügung steht, können den Kontextrouter dazu veranlassen, ihnen früher diese Information zukommen zu lassen. Auf diese Funktionalität wird im nächsten Abschnitt detaillierter eingegangen.

3.2.2 Solicitation

Benötigt ein Netzknoten in kürzester Zeit die Adresse des Kontextrouters, kann er diese mit einem Solicitation anfordern. Hierbei handelt es sich um ein sehr kurzes Paket (32 Bit). Wie Abbildung 3.3 zeigt, besteht das Paketformat im Wesentlichen aus dem Header des ICMP. Die Bedeutung der Felder *Type* und *Checksum* ist gleich der des oben beschriebenen Advertisement-Paketes. Des Weiteren gelten hier auch die dort beschriebenen Erweiterungen des Feldes *Code*. Der Wert des Feldes *Type* beträgt jetzt allerdings 10. Dem Header folgt ein 32 Bit langes Feld, was reserviert ist und mit Nullen aufgefüllt wird. Im Header des unter dem ICMP liegenden IP-Paketes befindet sich der Initiator der Solicitation-Nachricht als Quelladresse. Als Ziel wird hier auch wieder die *limited* Broadcast- bzw. die entsprechende Multicast-Adresse für ein Subnetz verwendet. Der Kontextrouter selbst soll sich nach dem Erhalt eines Solicitation-Paketes wie in RFC 1256 beschrieben verhalten. Danach antwortet er entweder mit einem Unicast direkt an den Initiator des Solicitations oder er sendet als Reaktion ein neues Advertisement.

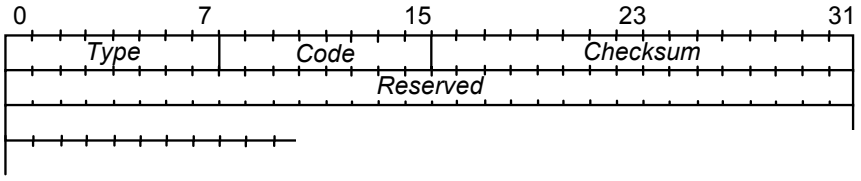


Abbildung 3.3: ICMP-Solicitation-Paket

3.2.3 Registrierung von kontextsensitiven Diensten

Ein Server, der kontextsensitive Dienste anbietet, soll diese auf dem Kontextrouter registrieren können. Eine Registrierung darf erst erfolgen, wenn sich der Server

authentifiziert hat. Hierfür sollte vorerst ein Challenge-Response-Verfahren verwendet werden, wie es beispielsweise bei GSM unter Nutzung des A3-Algorithmus zur Authentifizierung eingesetzt wird. Der Kontextrouter sendet dazu dem Server eine Zufallszahl. Der Server verschlüsselt mit dieser Zufallszahl sein Passwort und sendet das Ergebnis an den Kontextrouter zurück. Dieser wiederum kennt das Passwort und hat es seinerseits mit der Zufallszahl verschlüsselt. Der Kontextrouter kann nun beide Ergebnisse vergleichen. Stimmen diese überein, hat sich der Server korrekt authentifiziert. Die anschließende Kommunikation kann dann ebenfalls verschlüsselt erfolgen. Andere Varianten der Authentifizierung und Verschlüsselung sind ebenfalls denkbar. Zu erwähnen wäre hierbei beispielsweise der *Secure Sockets Layer* (SSL), auf den eine Reihe von Anwendungsprotokollen aufsetzen können. Nichtsdestotrotz existieren heutzutage mehrere bewährte Lösungen, die die Aufgaben dieses Kommunikationsteils sehr gut übernehmen können. Auf diese Varianten wird bei der späteren Implementierung zurückgegriffen werden.

Nach der erfolgreichen Anmeldung kann der Server die von ihm bereitgestellten kontextsensitiven Dienste und die zugehörigen Kontexttypen ablegen. Zur Übertragung der Daten wird auf herkömmliche Protokolle zurückgegriffen. Die Dienste und Kontexttypen werden dazu kodiert und können beispielsweise mit Hilfe von HTML, XML o. ä. übertragen werden. Wichtig ist an dieser Stelle, dass eine definierte Schnittstelle in der Routerarchitektur existiert, über die diese Daten abgelegt werden können. Als einfachste Variante könnte eine binäre Kodierung für die Übertragung der Daten genutzt. Dabei besitzt jeder Dienst und jeder Kontexttyp eine eindeutige binäre Kennung. Unabhängig davon, wie die Kodierung letztlich realisiert wird, ist sie unbedingt in Anlehnung an die Beschreibung des Dienstes und der Kontexttypen innerhalb eines RREQ vorzunehmen. Damit können diese Daten ohne großen Umrechnungsaufwand im Kontextrouter abgelegt und miteinander verglichen werden. Das bei der Übertragung vom Server zum Client verwendete Datenformat sollte auf jeden Fall erweiterbar sein. Dadurch wird es möglich, spätere zusätzliche als relevant angesehene Informationen, die der Entscheidungsfindung bei der Auswahl des Servers dienlich sein können, einzubinden.

Server	Dienste	Kontexttyp	IP-Adresse	Interface[1]	Hops[1]
S1	1	4, 5, 8, 7	129.187.39.54	2	9
S2	1	1, 2, 3	141.24.92.184	1	–
	27	7			
S3	1	1, 5, 6	141.24.93.161	1	–
	13	9			
	15	5, 7, 10, 14			
S4	1	–	141.24.92.61	1	–
S5	1	2, 3, 9, 10	141.30.2.2	3	10
	15	5, 7			
	29	12, 28, 36			
S6	1	1, 4	130.149.4.134	2	8

Tabelle 3.2: Beispiel für eine Server-Routing-Tabelle

[1]Diese Informationen können auch über die reguläre Routingtabelle bezogen werden.

Die übertragenen Daten werden dann in einer Tabelle auf dem Kontextrouter abgelegt. Tabelle 3.2 ist ein Vorschlag dafür, wie eine solche Tabelle aussehen könnte. Neben Dienst und zugehörigem Kontexttyp muss mindestens auch die IP-Adresse des Servers aufgelistet sein. Weitere Optionen wie Kosten, Hopanzahl (gerade beim Ad-hoc-Netz) usw. sind möglich und können später bei der Auswahl eines für den Nutzer passenden Servers als Entscheidungskriterium dienen. Nach der Registrierung muss sich der Server in regelmäßigen Abständen beim Kontextserver melden und seine Einträge erneuern. Erfolgt in der vorgegebenen Zeit kein Update, wird der Eintrag wieder aus der Tabelle gelöscht. Der Startzeitpunkt des Timers ist hierbei unabhängig von den anderen Servereinträgen und dem Senden von Advertisements. Damit wird gewährleistet, dass die Wahrscheinlichkeit von Verkehrsspitzen durch gleichzeitiges Senden vieler/aller Managementinformationen gering bleibt.

3.2.4 Anfrage nach kontextsensitiven Diensten

Die Anfrage nach einem kontextsensitiven Dienst erfolgt mit einem erweiterten AODV-RREQ vom Client aus. Erhält der Kontextrouter ein solches Paket, liest er die darin enthaltenen Informationen (Dienst, Kontexttypen, Priorität des jeweiligen Kontexttyps) aus. Im Anschluss daran muss verglichen werden, inwieweit der angeforderte Dienst mit den beim Router registrierten Diensten übereinstimmen. Über eine entsprechende Auswahllogik wird dann entschieden, welcher Server den geeignetsten Dienst anbietet. Hierfür legt der Kontextrouter eine interne Reihenfolge, eine „Serverrangliste“, für in Frage kommende Server fest. Optimalerweise sollte die Implementierung der Auswahllogik auf dem Router so gestaltet werden, dass sie auf bestimmte Anforderungen hin angepasst werden kann. Auf die Auswahllogik wird in Abschnitt 3.2.5 detaillierter eingegangen. Ist eine Auswahl getroffen worden, sendet der Router ein erweitertes AODV-RREP an den Client zurück, was entweder den passendsten Server mit den dort unterstützten Kontexttypen enthält oder die Anfrage negativ beantwortet. Eine Antwort ist dann negativ, wenn kein entsprechender Dienst auf dem Kontextserver registriert ist (siehe auch Abschnitt 3.4). Eine wiederholte Anfrage durch den Client ist dann möglich, wenn der Nutzer den vorgeschlagenen Server nicht in Anspruch nehmen will. Daraufhin kann der Kontextrouter auf zwei Arten reagieren. Haben sich die Kontexttypen und Prioritäten nicht geändert, wird der Identifikator ausgelesen. Damit ist klar, welches der aktuell ausgewählte Server war, und es wird als nächstes der Server vorgeschlagen, der im Auswahlprozess den nächsten Rang belegte. Handelt es sich um den letzten alternativen Server, wird das so genannte *No Alternatives*-Flag gesetzt, das mit dieser Routingarchitektur neu eingeführt wird. Wurden Kontexttypen oder Prioritäten geändert, muss ein neuer Auswahlprozess gestartet werden. Ein Vorschlag für den konkreten Aufbau der erweiterten RREQ-/RREP-Pakete erfolgt in Abschnitt 3.4.

Mit dem Antwortpaket sendet der Router auch den oben schon erwähnten Identifikator an den Client. Gleiche Identifikatoren können für verschiedene IP-Interfaces von Clients vergeben werden, müssen aber in der Kombination mit dem jeweiligen IP-Interface der Clients eineindeutig sein. Hier wird die IP-Adresse des ausgewählten Servers verwendet. Anhand dieses Identifikators kann eine spätere Zuordnung zwischen Client und Server erfolgen. Unproblematisch ist eine „parallele“ Kommunikation für zwei Dienste von einem Client aus zu ein und demselben Server, da die Anwendungskommunikation auf höheren Schichten erfolgt. Wird eine wiederholte Anfrage nach dem gleichen Dienst vom selben Client aus gesendet, so wird ihm

dieselbe Antwort wie bei einer Erstanfrage geschickt, da davon auszugehen ist, dass das vorhergehende RREP-Paket verloren gegangen ist.

Nachdem der Anfrage-/Antwortprozess beendet wurde, ist die Serverrangliste wieder zu löschen, zusätzlich hat sie eine begrenzte „Lebensdauer“. Erhält der Router ein RREQ ohne Kontexterweiterung hat er diese Anfrage wie ein herkömmliches Routingpaket zu behandeln. D. h., es wird auf das Paket mit einem RREP geantwortet, sofern die Route zum Ziel bekannt ist. Ist dies nicht der Fall, wird das RREQ-Paket mit einem Broad-/Multicast an die benachbarten Knoten weitergeleitet.

3.2.5 Serverauswahl

Die Auswahl eines kontextsensitiven Servers, der einer Anforderung eines Clients entspricht, kann auf unterschiedliche Weise erfolgen. So spielen neben dem vom jeweiligen Server unterstützten Dienst und den eigentlichen Kontexttypen auch deren vom Nutzer vergebenen Prioritäten eine Rolle. Der Einfluss der Prioritäten, die den Kontexttypen zugeordnet wurden, und verschiedene optionale Werte aus der Routingtabelle können während des Auswahlprozesses unterschiedlich berücksichtigt werden. So sind Kriterien wie Anzahl der Hops zum Server, Kosten und QoS-Unterstützung als zusätzliche Entscheidungsmerkmale möglich. Diese können auch miteinander kombiniert werden. Die Liste der Kriterien ist damit beliebig erweiterbar. Die Auswahlkriterien sind so vielfältig, dass eine spätere Implementierung der Auswahllogik sehr flexibel sein muss. Dem Provider muss es damit ermöglicht werden, diese entsprechend den Bedürfnissen des Nutzers anzupassen. Es sind auch Implementierungen denkbar, mit denen der Nutzer selbst über die gewünschte Auswahllogik entscheiden und diese auswählen kann. Eine weitere Herausforderung ist damit die Entwicklung allgemeingültiger Algorithmen sein, die eine spätere Erweiterung auf zusätzliche Auswahlkriterien auf einfache Weise ermöglichen sollen.

Das eigentliche Ziel des Auswahlalgorithmus besteht darin, nach einer Anfrage durch einen Client mittels erweitertem RREQ-Paket eine Rangfolge der in Frage kommenden Server zu erstellen. Diese Rangfolge soll dabei mit der Eignung der dort bereitgestellten Dienste für den Nutzer korrelieren. Der angeforderte Dienst gilt hierbei als Ausschlusskriterium. Dem Nutzer braucht kein Dienst angeboten werden, den er nicht angefordert hat. Daneben können die Kontexttypen und deren Prioritäten direkt zu einer Bewertung der Eignung eines Dienstes und damit zum Erstellen der Rangliste beitragen. Wie später noch beschrieben wird, erfolgt die Einteilung der Prioritäten in 8 Stufen. Dazu werden 3 Bit bereitgestellt. Eine feinere Granularität macht für den Nutzer wenig Sinn. Schließlich muss er diese Priorisierung vornehmen. Alternativ kann dies aber auch von einer Anwendung übernommen werden. Nichtsdestotrotz können schließlich verschiedene Kombinationen von Kriterien zu unterschiedlichen Ranglisten führen. Dies soll im Folgenden anhand von drei Beispielen verdeutlicht werden. Kontexttypen mit der Priorität 0 bilden eine Besonderheit. Damit sind solche Kontexttypen gemeint, die nicht von der Anwendung bzw. dem Nutzer gefordert sind aber zur Verfügung stehen. Solche Informationen können dann beispielsweise vom Server bzw. Provider für andere Dienste (z. B. Kontextinformationen über die Interessen des Nutzers für eine angepasste Werbung) gezielt genutzt werden.

Lineare Auswahl

Bei der einfachsten Form der Auswahl werden die Entscheidungskriterien nach ihrer Priorität abgearbeitet. Die Rangfolge der Server ergibt sich dann nach folgendem Schema, was durch Abbildung 3.4 noch einmal veranschaulicht wird. Auf eine mathematische Darstellung des Auswahlverfahrens wird an dieser Stelle verzichtet, da es sich hier um einen einfachen Größenvergleich zwischen verschiedenen Werten handelt. Der Algorithmus arbeitet dann wie folgt:

1. Zuerst wird eine Reihenfolge für die geforderten Kontexttypen eines jeden Servers, die den geforderten Dienst anbieten, gebildet. Diese beginnt jeweils mit dem Kontexttyp der höchsten Priorität und endet mit dem der kleinsten Priorität.

2. Server, deren unterstützte Kontexttypen mit den gleichen Prioritäten beginnen, bilden eine gemeinsame Gruppe. Von den so entstandenen Gruppen wird wieder eine Rangfolge erstellt. Die Gruppe mit der höchsten Priorität steht an oberster Stelle in der Rangliste. Danach kommt die Gruppe mit den Servern, deren Kontexttypen die nächst kleinere Priorität besitzen usw.

3. Jetzt erfolgt die gleiche Vorgehensweise für jede einzelne Gruppe, sofern dort mehr als ein Server enthalten ist. Dabei müssen jedoch die zuletzt verglichenen Prioritäten unberücksichtigt bleiben. Es wird in die Gruppe hineingegangen und wieder Punkt 2 abgearbeitet. Damit erhält man eine Rangfolge innerhalb einer jeden Gruppe. Zu beachten ist hierbei, dass Server bzw. deren Dienste, die keine weiteren Kontettypen unterstützen, den Wert Null erhalten und somit am Ende der Rangliste innerhalb der aktuell betrachteten Gruppe einnehmen. Die Vorgehensweise wird dann so oft wiederholt, bis nur einzelne oder vollkommen gleichwertige Server übrig bleiben.

4. Am Ende sind alle Server in eine Rangfolge eingeordnet. Tritt der Fall auf, dass zwei gleichwertige Server eine gemeinsamen Rang einnehmen, können zusätzliche Entscheidungsmerkmale (z. B. Hopanzahl zum Server, Lastverteilung, ...) verwendet werden, um eine optimale Serverauswahl treffen zu können.

Nachteilig bei dem oben beschriebenen Auswahlverfahren ist, dass nur die Wertigkeiten der Prioritäten nicht aber die Menge der unterstützten Kontexttypen berücksichtigt werden. Somit wird beispielsweise ein Server bevorzugt, der vielleicht nur einen einzigen Kontexttyp, welcher aber die höchsten Priorität besitzt, unterstützt. Ein Server, der eine Vielzahl von geforderten Kontexttypen bereitstellt, denen aber eine kleinere Priorität zugeordnet ist, wird dann erst dahinter in der Rangliste aufgenommen. Ein Verfahren, was die Anzahl der unterstützten Kontexttypen mit berücksichtigt, wird im folgenden Abschnitt beschrieben.

Kumulierte Auswahl

Bei diesem Auswahlalgorithmus wird versucht, neben den Prioritäten auch die Anzahl der bei den einzelnen Servern zur Verfügung stehenden Kontexttypen zu berücksichtigen. Mit anderen Worten – es wird ein Maßstab zum Ausdruck der mittleren

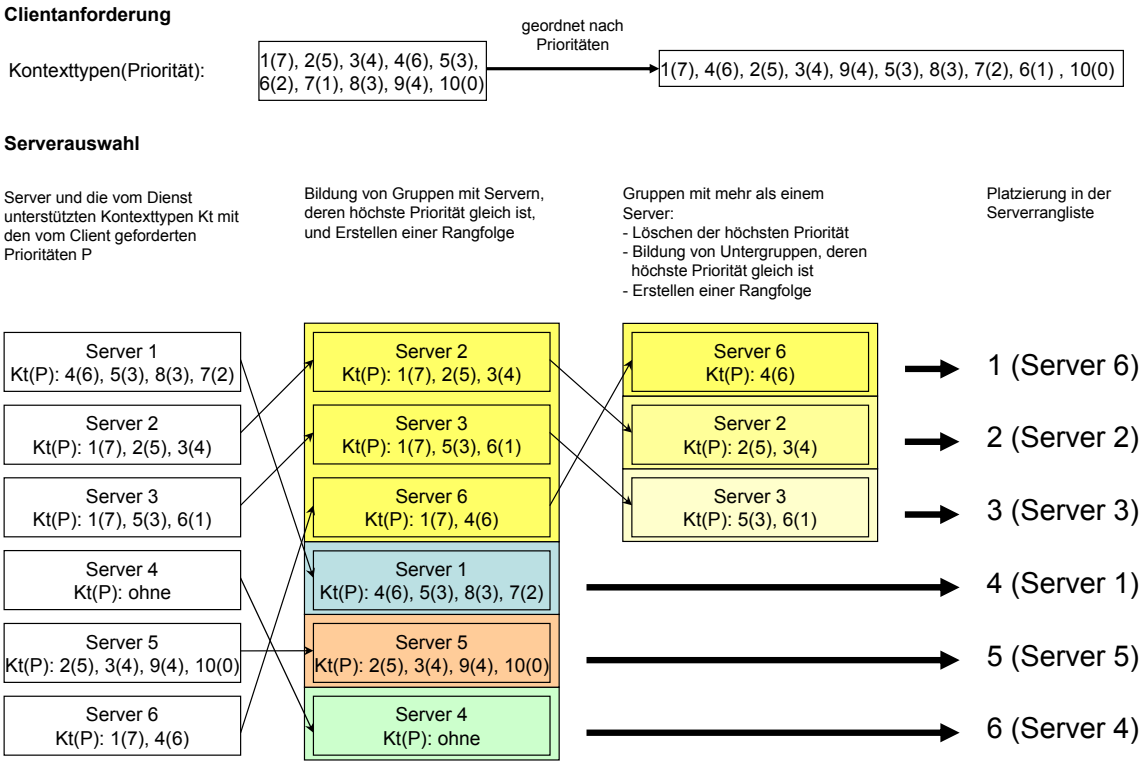


Abbildung 3.4: Entscheidungsalgorithmus für die lineare Auswahl

Kontextdichte gesucht. Dazu kann zum Beispiel ein prozentualer Vergleich durchgeführt werden. Ein Server, der einen Dienst anbietet, dessen unterstützten Kontexttypen und deren Prioritäten genau den geforderten des Clients entsprechen, erhält damit 100%. Die Werte für die anderen Server ergeben sich damit aus:

$$X_k = \frac{\{\sum P(KtS)\}_k}{\sum P(KtC)} \times 100\% \tag{3.1}$$

X_k ist damit ein prozentualer Wert, der ausdrückt inwieweit der Dienst des Servers k mit der Anforderung des Clients bezüglich der Kontexttypen übereinstimmt. Der Zähler ergibt sich aus der Summe der zu diesem Server gehörenden Prioritäten $P(KtS)$. Der Nenner setzt sich aus der Summe der Prioritäten der vom Client geforderten Kontexttypen KtC zusammen. Damit ergibt sich nun die in Tabelle 3.3 dargestellte Serverrangliste. Verbleiben im Ergebnis einmal gleichrangige Server, wird wie in Abschnitt 3.2.5 unter Punkt 4 des Auswahlalgorithmus verfahren.

Wie in Tabelle 3.3 auch zu sehen ist, werden jetzt zwar alle Kontexttypen berücksichtigt, allerdings ist hier keine Möglichkeit gegeben, die einzelnen Prioritäten unterschiedlich zu wichten. Diese Möglichkeit wird im nächsten Abschnitt dargestellt.

Gewichtete Auswahl

Um höheren Prioritäten auch bei der Auswahlentscheidung eine größere Wichtung zukommen lassen zu können, muss die Gleichung 3.1 modifiziert werden. Dazu eignet sich beispielsweise das Potenzieren der Prioritäten. Je größer die Wahl der Exponenten ist, desto höher wird der Einfluss großer Prioritäten. Natürlich kann auch mit

Server	Kontexttyp(Priorität)	X_k in %	Platzierung
S1	4(6), 5(3), 8(3), 7(2)	40,0	2
S2	1(7), 2(5), 3(4)	45,7	1
S3	1(7), 5(3), 6(1)	31,4	5
S4	–	0	6
S5	2(5), 3(4), 9(4), 10(0)	37,1	3
S6	1(7), 4(6)	37,1	3

Tabelle 3.3: Serverrangliste nach kumulierter Auswahl

unterschiedlichen Exponenten gearbeitet werden. Die folgende Gleichung zeigt ein Beispiel für die Nutzung eines einheitlichen Exponenten:

$$X_k = \frac{\left\{ \sum_{i=0}^{N} [P_i(KtS)]^j \right\}_k}{\sum_{i=0}^{M} [P_i(KtC)]^j} \times 100\% \qquad \text{mit j>0} \tag{3.2}$$

Dabei stellt der prozentuale Wert X_k wiederum einen Vergleichswert für den Dienst eines Servers k in Bezug auf die Übereinstimmung zum geforderten Dienst und den entsprechenden Kontexttypen dar. Allerdings wird hierbei im Gegensatz zu Gleichung 3.1 die unterschiedliche Wichtung der Prioritäten berücksichtigt. Dazu werden die Prioritäten P der einzelnen Kontexttypen für den betrachteten Server k mit j potenziert und anschließend summiert. Der Wert j stellt dabei ein Maß für die Wichtung der Prioritäten dar. Er muss vor dem Auswahlprozess festgelegt werden. Je höher j gewählt wird, desto größer ist die Wichtung hoher Prioritäten. N ist die Gesamtzahl der unterstützten Kontexttypen, die mit den vom Client geforderten Typen übereinstimmen. M ist die Gesamtzahl der vom Client gelieferten Kontexttypen. Bei gleichwertigen Servern, d. h. die Werte X_k sind gleich, wird wiederum wie in Abschnitt 3.2.5 unter Punkt 4 der Auswahllogik verfahren. Tabelle 3.4 zeigt, welches Ergebnis sich nun für das obige Beispiel ergeben würde. Zwar ist auch hier der erste Platz durch Server 2 belegt, allerdings belegt nun Server 6 den zweiten Rang, da dies der einzige Server ist, dessen Dienst die zwei Kontexttypen mit den beiden höchsten Prioritäten unterstützt. Obwohl andere Server mehr Kontexttypen unterstützen, erfolgt durch die geringere Wichtung der Prioritäten eine Abwertung.

Server	Kontexttyp(Priorität)	X_k in %	Platzierung
S1	4(6), 5(3), 8(3), 7(2)	35,2	4
S2	1(7), 2(5), 3(4)	54,5	1
S3	1(7), 5(3), 6(1)	35,8	3
S4	–	0	6
S5	2(5), 3(4), 9(4), 10(0)	34,5	5
S6	1(7), 4(6)	51,5	2

Tabelle 3.4: Serverrangliste nach gewichteter Auswahl

Sicherlich gibt es noch eine Vielzahl von Auswahlmöglichkeiten. Hinzu kommt, dass die Algorithmen miteinander kombiniert werden können. Bisher wurden auch die zusätzlichen Routinginformationen nur ansatzweise zur Entscheidungsfindung erwähnt. Letztlich muss die Praxis zeigen, welches Verfahren am geeignetsten ist. Gerade weil die Möglichkeiten an dieser Stelle so vielfältig sind, sollte bei einer späteren Realisierung unbedingt darauf geachtet werden, dass die Implementierung des Auswahlverfahrens flexibel erfolgt. Damit kann gewährleistet werden, dass ein Austausch oder eine Modifikation des Entscheidungsalgorithmus leicht und unabhängig von anderen Bestandteilen der Routingarchitektur erfolgt. Es wäre auch möglich, auf dem Kontextrouter mehrere Algorithmen zu implementieren. Dann könnte auch, wie bereits erwähnt, der Nutzer bzw. die kontextsensitive Anwendung bestimmen, welcher davon zum Finden des geeigneten Servers genutzt werden soll. Ist schließlich ein Server ausgewählt worden, wird dies dem Client mitgeteilt. Wie die daran anschließende Kommunikation mit dem Server aussehen kann, beschreibt der folgende Abschnitt.

3.2.6 Weiterleitungsfunktion

Hat der Client die Antwort des Kontextrouters empfangen und den ausgewählten Server akzeptiert, kann er die Kommunikation mit ihm initiieren. Wie in den Abschnitt 1 beschrieben, wird dafür eine indirekte Kommunikation über den Kontextrouter initiiert. Der Client sendet dazu seine Daten weiterhin an den Kontextrouter und dieser leitet sie an den entsprechenden Server weiter. Abbildung 3.5 beschreibt dieses Szenario. Empfängt der Kontextrouter ein IP-Paket, ordnet er dies anhand der Quelladresse und des mit dem Client für diese Kommunikation vereinbarten Identifikators dem gewünschten Zielserver zu. Danach wird die Adresse des IP-Paketes ausgetauscht. Das Paket erhält nun als Ziel die Adresse des Servers. Der Identifikator wird ebenfalls ersetzt. Als Identifikator wird nun die IP-Adresse des Clients verwendet. Mit diesem Identifikator werden schließlich die vom Server zurückgesendeten Pakete eindeutig dem entsprechenden Client zugeordnet. Erhält der Kontextrouter ein solches IP-Paket von einem Server, wird dieses nun direkt an den Client weitergeleitet. Erhält der Kontextrouter keine Antwort in einer bestimmten Zeit, sendet er eine Meldung an den Client (z. B. mit einer entsprechenden ICMP-Nachricht) zurück und löscht alle zu diesem Server gehörenden Einträge in seinen Tabellen. Alternativ kann der Kontextrouter die folgende Kommunikation auch auf einen gleichwertigen Server, d. h. der den gleichen Dienst mit denselben Kontexttypen unterstützt, umleiten. Problematisch ist an dieser Stelle, dass die bis dahin empfangenen Informationen und aufgenommenen Kontextdaten verloren sind, wie in [Renh06] schon richtig festgestellt wurde. Bei einigen kontextsensitiven Diensten könnte dies zu Problemen führen. Hat der Nutzer beispielsweise einen Navigationsdienst benutzt, weiß der alternative Server zwar die aktuelle Position des Nutzers, die gewählte Route ist ihm aber unbekannt. Da es von der Anwendung abhängig ist, ob die „historischen“ Daten benötigt werden, sollte diese Entscheidung dem Client überlassen werden. So kann dieser schon mit dem Senden des erweiterten RREQ durch Setzen eines Flagbits angeben, ob bei Ausfall eines Servers automatisch auf einen alternativen Server umgeleitet werden soll. Dieses Flag wird in späteren Abschnitten als *Non Reroute*-Flag bezeichnet. Ist es gesetzt, wird keine automatische Weiterleitung auf einen alternativen Server gewünscht. Aus Abbildung 2.1 ist bekannt, dass im Paketkopf der RREQ- und RREP-Pakete reservierte Bereiche vorhanden sind. Davon kann ein Bit für das so genannte *No Reroute*-Flag verwendet. Es reicht, dieses Flag innerhalb der

RREQ-Pakete vorzusehen. Für die Realisierung des Konzeptes wird vorgeschlagen, das erste freie Bit nach den regulären Flags im Paketkopf zu verwenden.

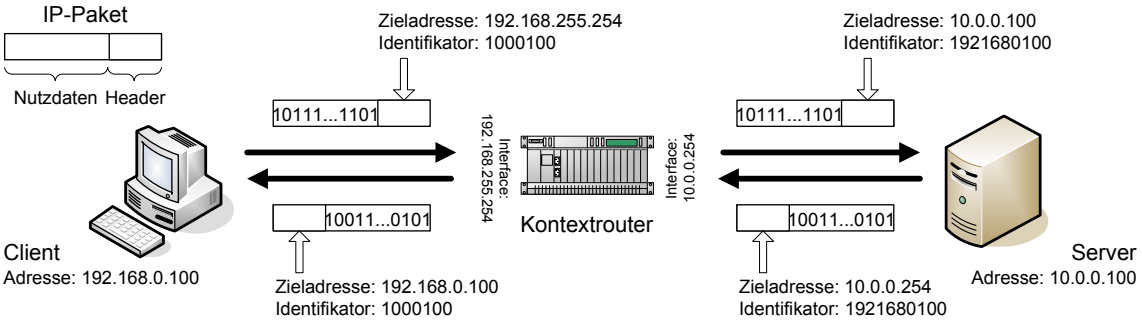


Abbildung 3.5: Weiterleitung durch den Kontextrouter

Erhält der Kontextrouter ein (ungültiges) IP-Paket, welches zwar an ihn adressiert ist, er aber keine Zuordnung zwischen Absender-IP-Adresse und Identifikator besitzt, muss dieses Paket verworfen werden. Das gilt sowohl für Pakete, die vom Client als auch für solche die vom Server gesendet wurden. Eine Reaktion des Kontextrouters beispielsweise über ICMP ist nicht notwendig. IP-Pakete, deren Optionen keinen Identifikator enthalten, werden als reguläre Daten wie bei einem herkömmlichen Router weitergeleitet.

Für die Übertragung des Identifikators für die Kommunikation wird sowohl auf der Teilstrecke Client–Router als auch auf der Teilstrecke Router–Server das beim Internetprotokoll vorgesehene Feld für Optionen genutzt. Beim IPv6 können solche Optionen durch die Nutzung eines *Extended Headers* angegeben werden. Dort sind dann natürlich jeweils die Vorgaben aus den entsprechenden Spezifikationen zu berücksichtigen. Wenn die IP-Adressen, wie hier vorgeschlagen, als Identifikatoren verwendet werden, erübrigt sich eine Kontrolle für eine doppelte Vergabe. Die Zuordnungen sind dann eindeutig. In Abbildung 3.6 ist der Aufbau des *Options*-Feldes für das IPv4 dargestellt. Verpflichtend ist hierbei das erste Byte für den *Option-Type*. Dieses ist wiederum in drei Felder gegliedert und folgt direkt der Ziel-Adresse (Feld: *Destination Address*) im IP-Paketheader. Zum besseren Verständnis der späteren Verwendung sollen die Felder kurz erläutert werden.

Copied Flag (C) - wird gesetzt, wenn die Option bei einer Fragmentierung in jedes Teilfragment kopiert werden soll.

Option Class (OC) - Folgende Klassen sind vergeben: 0 = *Control* (Steuerung); 2 = *Debbuging and Measurement* (Fehlersuche und Messen); 1 und 3 sind reserviert für zukünftige Anwendungen.

Option Number (ON) - Die Werte 0–4 und 7–9 sind bereits für folgende Optionen vergeben: *End of Operation, No Operation, Security, Loose Source Routing, Strict Source Routing, Record Route Stream ID* und *Internet Timestamp.*

Ein weiteres Byte ist nötig, sobald zusätzlich auch Bereiche zur Übertragung weiterer Optionsdaten (*Option-Data*) benötigt werden. Das zugehörige Feld heißt *Option-Length* und gibt die Gesamtzahl aller für die Optionen verwendeten Bytes an. Das

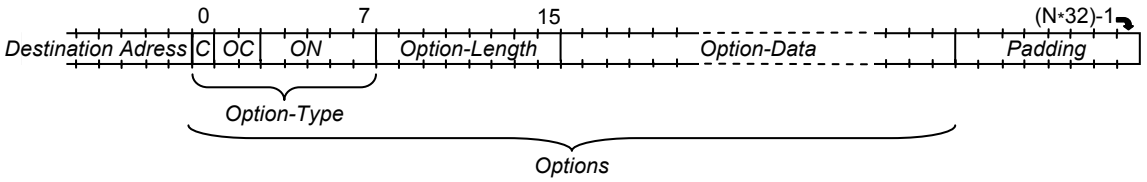


Abbildung 3.6: Das *Options*-Feld im IPv4

Padding-Feld ist dann so aufzufüllen, dass der gesamte IP-Paketkopf ein Vielfaches von 32 ergibt.

Abbildung 3.7 zeigt einen Vorschlag für die Implementierung des Optionen-Feldes innerhalb der Architektur für das kontextsensitive Routing. Das *Copied Flag* ist gesetzt, damit die Optionen eines jeden Teils eines fragmentierten IP-Pakets vom Kontextrouter ausgewertet werden können. Das Feld für *Option Class* kann auf 0 gesetzt werden, da es sich beim kontextsensitiven Routing um eine Steuerungsfunktion handelt. Die *Option Number* kann bis auf die vergebenen Werte (0-4 und 7-9) frei gewählt werden. Hier wurde 16 gewählt und dem *Context Sensitive Routing* zugeordnet. Das Längenfeld hat den Wert 4, da davon ausgegangen wird, dass 4 Byte für die Vergabe von Identifikatoren (*Identifier*) ausreichen. Insgesamt werden also 6 Byte benötigt, sodass das *Padding*-Feld eine Größe von 2 Byte besitzen muss. Für IPv6 muss die Formatierung entsprechend angepasst werden, was zum einen die Länge des Identifikators (128 Bit) und zum anderen die Einbindung in das Paket betrifft.

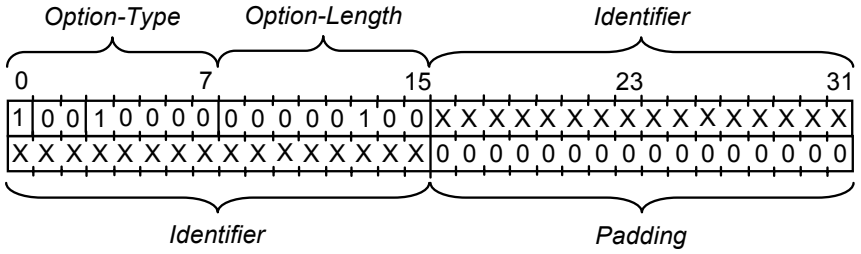


Abbildung 3.7: Belegung der Felder *Options* und *Padding* (IPv4)

Für die Kommunikation zwischen Kontextrouter und kontextsensitivem Server könnte man die eindeutige Zuordnung der Daten auch mit Hilfe des Network Address Translation (NAT) realisieren. Allerdings würde hierbei auf Protokolle höherer Schichten wie TCP/UDP, die sich in der Transportschicht befinden, zurückgegriffen. Dies erhöht die Rechenlast und Komplexität des Routers, weil damit zwischen Server und Client keine direkte Ende-zu-Ende-Kommunikation auf der Transportschicht möglich ist. Mit der hier vorgestellten Lösung und der Verwendung von Identifikatoren auf der Vermittlungsschicht werden auch die von NAT verursachten Probleme (siehe RFC 3027 [SrHo01]) vermieden. Das kontextsensitive Routing ist also lediglich auf die Netzfunktionen beschränkt. Für die höheren Protokollschichten, die Ende-zu-Ende-Funktionen erfüllen, ist es vollkommen transparent.

Wie bereits erwähnt, wäre auch eine direkte Kommunikation zwischen Client und Server unabhängig vom Kontextrouter prinzipiell mit dem hier vorgeschlagenen Konzept möglich. Dabei würde auf die Weiterleitungsfunktion eines Routers verzichtet. Es wäre dann auch kein Identifikator im RREP-Paket mehr nötig. Dem Client würde

dafür im RREP zusätzlich die Zieladresse des ausgewählten Servers mitgeteilt. Damit könnten dann Client und Server direkt miteinander kommunizieren – allerdings mit den in Abschnitt 1 aufgezählten Nachteilen.

Das hier vorgestellte Modell des Kontextrouters kann natürlich noch erweitert werden. So ist es beispielsweise vorstellbar, dass auch eine Liste alternativer Kontextrouter vorgehalten wird. Diese können wiederum ihr Wissen über kontextsensitive Server austauschen. Damit wäre es dann auch möglich, Anfragen, die von einem Kontextrouter nicht beantwortet werden können, an einen anderen Kontextrouter weiterzuleiten. Daneben sind auch Strategien zur Last- und Funktionsteilung denkbar. Die Liste könnte noch beliebig fortgeführt werden, soll aber Thema zukünftiger Forschungsarbeiten sein.

3.3 Kontextsensitiver Server

Nachdem die Funktionen des Kontextrouters festgelegt wurden, definiert dieser Abschnitt die Aufgaben, die der kontextsensitiven Server mindestens zu erfüllen hat, um in die kontextsensitive Routingarchitektur eingebunden zu werden. Der Server muss die von einem Kontextrouter empfangenen Advertisement-Pakete auswerten können. Die so erhaltenen IP-Adressen werden abgespeichert. Sie bleiben für eine bestimmte Zeit gültig. Nach RFC 1256 wären das beispielsweise 30 Minuten. Erhält der Server innerhalb dieser Zeit keine weiteren Advertisements von einem Router, ist dieser aus der Liste der bekannten Kontextrouter wieder zu entfernen. Besitzt der Server keine Adresseinträge (z.B. nach einem Neustart), kann er durch Senden von Solicitation-Paketen eventuell im Netz vorhandene Kontextrouter dazu veranlassen, ihm ihre IP-Adresse mitzuteilen. Zu beachten ist, dass Solicitations auch wiederholt gesendet werden dürfen, sofern innerhalb einer bestimmten Zeit keine Reaktion in Form einer Antwort durch einen Kontextrouter erfolgte. Allerdings ist die Zahl der Solicitation-Pakete, die gesendet werden dürfen, begrenzt. Im Wesentlichen sollte sich bei der Umsetzung des Verhaltens beim Empfang von Advertisement-Paketen und der zu realisierenden Solicitations an den RFC 1256 gehalten werden. Zusätzlich ist vorzusehen, dass dem Server die Adressen von kontextsensitiven Routern auch manuell übergeben werden können.

Der Server kann nun auf allen ihn bekannten Kontextroutern seine kontextsensitiven Dienste und die von ihnen unterstützten Kontexttypen registrieren, sofern er dort die entsprechenden Zugangsrechte besitzt. Dazu muss er sich beim Kontextrouter authentifizieren. Wie bereits in Abschnitt 3.2.3 beschrieben, können die Authentifizierung und die anschließende Kommunikation zur Registrierung der Dienste und Kontexttypen über herkömmliche Protokolle erfolgen. Eine Modifikation ist nicht nötig. Der Server muss dem Kontextrouter seine Anwesenheit im Netzwerk in definierten Abständen mitteilen, da sonst die dort gespeicherten Einträge gelöscht werden. Die Anwesenheitsmitteilung sollte auch die aktuell angebotenen Dienste und die zugehörigen Kontexttypen enthalten. Dadurch können die Daten auf dem Kontextrouter aktuell gehalten werden, selbst wenn zwischenzeitlich Dienste auf dem Server aus- bzw. wegfallen oder hinzukommen.

Empfängt der Server eine Anfrage nach einen kontextsensitiven Dienst und erhält er mit dem zugehörigen empfangenen IP-Paket einen Identifikator, so muss er diesen beim Antworten der Anfrage in das zu sendende IP-Paket kopieren. Prinzipiell

kann der Server auch Anfragen von Netzknoten beantworten, die ihm nicht als Kontextrouter bekannt sind. Die Kommunikation mit der kontextsensitiven Anwendung selbst wird oberhalb der Vermittlungsschicht erbracht und ist nicht Bestandteil dieser Arbeit.

3.4 Client

Dieser Abschnitt beschreibt die Funktionen, die mindestens vom Client erbracht werden müssen, um die Dienste der kontextsensitiven Routingarchitektur in Anspruch nehmen zu können. Die hier vorgeschlagene Beschreibung der Dienste innerhalb der erweiterten RREQ- und RREP-Pakete ist dabei sehr einfach gehalten. Zum Verifizieren der Funktionen der Routingarchitektur soll die hier verwendete Variante aber vorerst ausreichen. Bei späteren Forschungen ist jedoch zu untersuchen, wie die Beschreibung der Dienste weiter optimiert werden kann.

Bevor der Client ein erweitertes AODV-RREQ-Paket zur Anfrage nach einem kontextsensitiven Dienst senden kann, benötigt er die Adresse eines Kontextrouters. Um diese zu erhalten, können prinzipiell die gleichen Mechanismen, wie sie im Abschnitt 3.3 beschrieben wurden, verwendet werden. Allerdings stellt hier der manuelle Eintrag der IP-Adresse des kontextsensitiven Servers eine Ausnahme dar. Auf diese Möglichkeit kann ganz verzichtet werden, da sich die Clients im gleichen Netz befinden wie die Kontextrouter. Dies wurde bereits in Abschnitt 3.1 begründet und ist auf das in dieser Arbeit verfolgte Konzept zurückzuführen.

Um einen kontextsensitiven Dienst zu nutzen, sendet der Client ein erweitertes AODV-RREQ an einen Kontextrouter. Dazu muss im RREQ-Header das *D*-Flag gesetzt werden. Unter Nutzung des IPv6 wird beim dort verwendeten AODV die *Flood Data Option* verwendet. Beides ist notwendig, damit nur das eigentliche Ziel – nämlich der Kontextrouter – auf eine Anfrage antwortet. Die Angabe des kontextsensitiven Dienstes erfolgt in den Erweiterung des AODV-Paketes. Ein Vorschlag, wie diese Erweiterung genutzt werden kann, ist in Abbildung 3.8 dargestellt. Dabei bleibt das Format aus RFC 3561 erhalten. Im Feld *Type* wird gekennzeichnet, dass es sich um eine Anfrage nach einem kontextsensitiven Router handelt. Der Wert 1 ist schon für das *Hello Interval Extension Format* vergeben, deshalb kann für das das vorliegende Szenario beispielsweise 2 verwendet werden. Mit dem Datenteil werden schließlich der Identifikator, der angeforderte Dienst, die Kontexttypen und deren Priorisierung übertragen. Dazu wird vorerst eine relativ einfache Codierung vorgeschlagen. Die ersten 4 Bytes dienen zur Übertragung des Identifikators. Dort wird der für die aktuelle Dienstanfrage zuletzt vom Kontextrouter erhaltene Identifikator abgelegt. Das erste RREQ erhält eine 0. Die darauf folgenden 2 Bytes, beschreiben den angeforderten Dienst (*Service*). Insgesamt können somit 65535 Dienste unterschieden werden. Von den verbleibenden 253 Bytes werden vorerst 50 Bytes genutzt, um die Kontexttypen (*Context-Types (Ct)*) zu übertragen. Zu jedem Kontexttyp wird auch dessen Priorität angegeben. Zur Priorisierung sind 3 Bit, in Abbildung 3.8 als *P* gekennzeichnet, vorgesehen. Damit sind 7 Prioritätsstufen möglich, wobei die Priorität mit dem Wert steigt (0 = keine Priorität; 7 = höchste Priorität). Ein Bit wird als Flag (*C*) verwendet, um zu kennzeichnen, dass der jeweilige Kontexttyp gewählt wurde. Damit werden pro Byte zwei Kontexttypen und deren Priorität beschrieben. Insgesamt können also 100 Kontexttypen kombiniert übertragen werden. Das Längenfeld (*Length*) erhält somit den Wert 56, die Erweiterung ist insgesamt 58

Bytes groß. Natürlich kann dieses Format noch bis auf die maximal mögliche Größe erweitert werden. Das hängt von den später tatsächlich definierten bzw. verwendeten Kontexttypen ab. Andererseits können auch beispielsweise Prioritäten für Optionen angegeben werden, die dem Auswahlprozess im Abschnitt 3.2.5 dienlich sind.

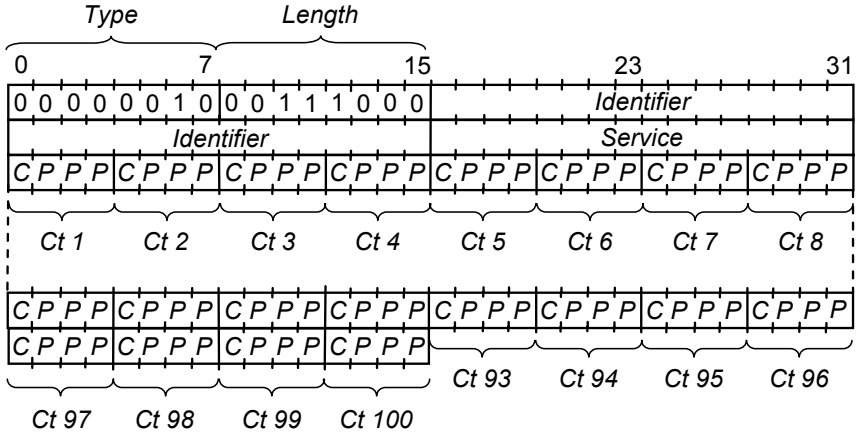


Abbildung 3.8: Format des AODV-RREQ mit Kontexterweiterung

Nach dem Versenden des erweiterten AODV-RREQ wartet der Client eine definierte Zeit (Timeout) auf das zugehörige RREP. Kommt keine Antwort zurück, kann die Anfrage mit einer endlichen Anzahl von Versuchen wiederholt werden. Bleibt dies erfolglos, gilt der Kontextrouter als nicht erreichbar. Die Informationen über ihn werden gelöscht. Entweder ist nun ein alternativer Kontextrouter zu wählen oder es wird ein Solicitation-Paket gesendet, um neue Kontextrouter zu finden. Bleiben beide Varianten erfolglos, ist kein kontextsensitives Routing möglich. Es erfolgt dann eine herkömmliche Dienstanfrage ohne kontextsensitives Routingverfahren, sofern dies die kontextsensitive Anwendung zulässt.

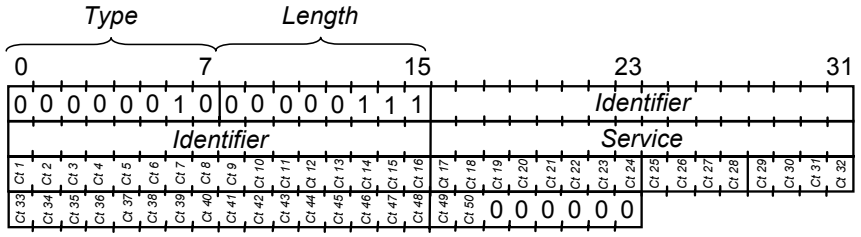


Abbildung 3.9: Format des AODV-RREP mit Kontexterweiterung

Erhält der Client eine Antwort auf das erweiterte RREQ, erfolgt diese in Form eines erweiterten RREP. D. h., es wird bei dem Paketformat ebenfalls die Möglichkeit der Paketerweiterung genutzt. Wie in Abbildung 3.9 zu sehen ist, werden dort der Identifikator (*Identifier*), der Dienst (*Service*) und die vom Server unterstützten Kontexttypen (*Kt*) angegeben. Das Format der Erweiterung ist dem des RREQ-Paketes angelehnt. Das Feld *Type* ist gleich. Vom Datenfeld werden die ersten 4 Bytes für den Identifikator verwendet. Danach folgen 2 Bytes für den Dienst und 7 Bytes für die Kontexttypen. Damit erhält das Feld *Length* den Wert 13. Insgesamt hat die Erweiterung also eine Größe von 15 Bytes. Die freien Stellen des letzten Bytes werden mit Nullen aufgefüllt, da das Format nur ganzzahlige Bytegrößen zulässt. Zusätzlich befindet sich im reservierten Bereich des Paketheaders das *No Alternatives*-Flag,

was auf 0 gesetzt ist, solange bei der Serverwahl nicht das Ende der Serverrangliste erreicht ist. Es wird vorgeschlagen, die erste Position hinter den regulären Flags im Paketheader für das *No Alternatives*-Flag zu verwenden.

Der Identifikator dient dem Kontextrouter der eindeutigen Zuordnung bei der späteren Weiterleitung der eigentlichen Daten. Da es sich dabei um die IP-Adresse des ausgewählten Servers handelt, kann der Nutzer bzw. Client entscheiden (z. B. anhand einer „Schwarzen Liste“), ob er diesen Server akzeptiert. Zusätzlich fließen in diesen Entscheidungsprozess auch die mitgelieferten Kontexttypen ein. Hierbei handelt es sich um die Typen, die von dem ausgewählten Server und damit den dort angebotenen Dienst auch tatsächlich unterstützt werden. Daraus ergeben sich folgende Möglichkeiten einer Reaktion durch den Client:

1. Der Client lehnt den Server ab und stellt keine neue Anfrage an den Kontextrouter.

2. Der Client lehnt den Server ab und stellt eine neue Anfrage an den Kontextrouter. In der neuen Anfrage wird der Identifikator des letzten RREP mitgesendet, sodass als Antwort ein alternativer Server zu erwarten ist. Diese Varaiante ist nur möglich, wenn kein *No Alternatives*-Flag im RREP gesetzt war.

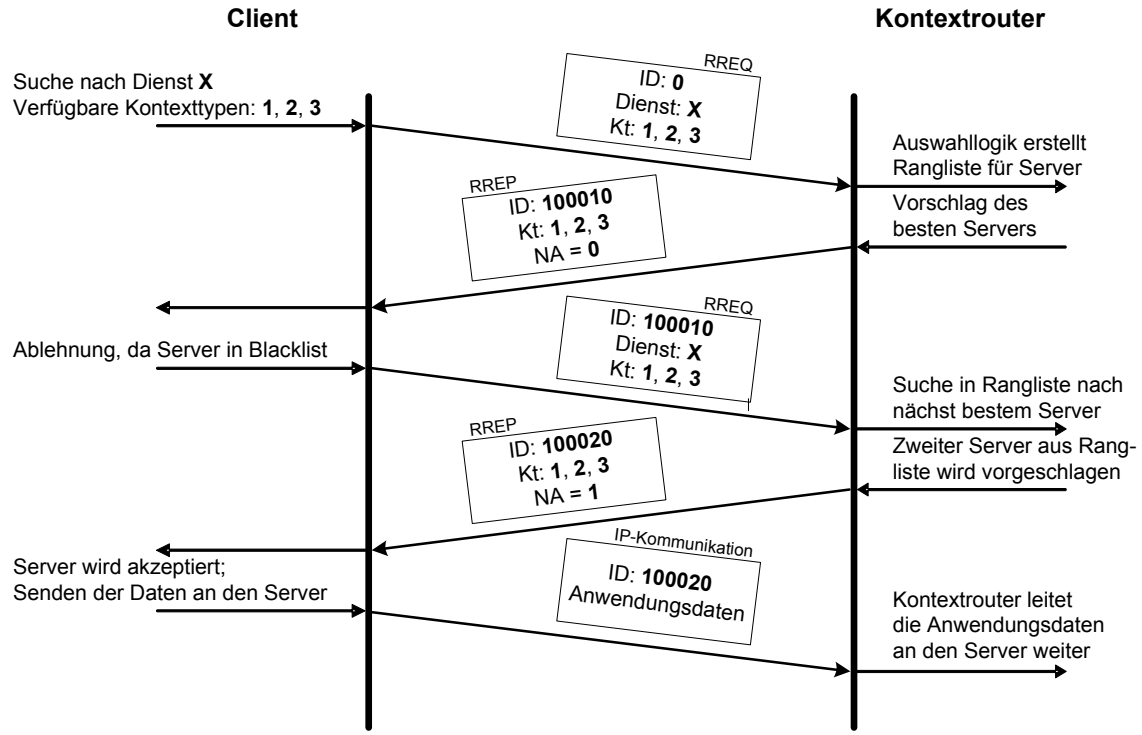
3. Der Client lehnt den Server ab und stellt eine neue Anfrage an den Kontextrouter, wobei die Kontexttypen oder deren Prioritäten in der neuen Anfrage modifiziert wurden. Es spielt hierbei keine Rolle, welchen Wert der einzusetzende Identifikator erhält.

4. Der Client akzeptiert den Server. Er verwendet den mitgelieferten Identifikator für die anschließende Kommunikation.

Als Beispiel zeigt Abbildung 3.10 ein Message Sequence Chart für den Fall, dass der vom Kontextrouter vorgeschlagenen Server abgelehnt wurde. Daraufhin schlägt der Kontextrouter einen altenativen Server vor, setzt aber das *No Alternatives*-Flag, da das Ende der Rangliste erreicht ist. Der Client akzeptiert den zweiten Server und beginnt mit diesem zu kommunizieren.

Die Anfrage nach einem anderen Dienst ist unabhängig von der aktuellen Entscheidung zu jedem Zeitpunkt möglich, da vom Client durch die Angabe des Dienstes innerhalb des RREP eine eindeutige Zuordnung erfolgen kann. Erhält der Client ein RREP-Paket, bei dem eines der Felder für den Identifikator oder den Dienst den Wert 0 hat, wurde die Anfrage vom Kontextrouter abgelehnt.

Hat der Client schließlich einen Server akzeptiert, kann die eigentliche Kommunikation mit diesem beginnen. Dazu werden die IP-Pakete entsprechend Abschnitt 3.2.6 präpariert. Der Identifikator wird aus dem letzten RREP-Paket übernommen. Fällt während der Kommunikation der Server aus, kann der Kontextrouter die Kommunikation automatisch auf einen gleichwertigen Server umleiten. Wie im eben erwähnten Abschnitt aber auch schon beschrieben wurde, ist dies nicht in jedem Fall sinnvoll. Deshalb kann der Client beim Senden des erweiterten RREQ das bereits in Abschnitt 3.2.6 beschriebene *No Reroute*-Flag setzen und diesen Automatismus vermeiden.



Rangliste der geeigneten Server

Server	Kontexttypen
10.0.0.10	Kt **1**, Kt **2**, Kt **3**
10.0.0.20	Kt **1**, Kt **2**

ID: Identifikator
Kt: Kontexttyp
NA: *No Alternatives*-Flag

Abbildung 3.10: Auswahl eines Servers durch den Client

Schließlich unterliegt es dem Client zu entscheiden, ob eine aktuelle Anfrage durch eine Anwendung überhaupt einen kontextsensitiven Dienst verlangt. Dazu müssen der Client bzw. die dort implementierte Software für das kontextsensitive Routing eine geeignete Schnittstelle bereitstellen, an der diese Informationen übergeben werden können. Damit ist der Client mit den in diesem Abschnitt unterbreiteten Vorschlägen in der Lage, einen kontextsensitiven Dienst mit Hilfe der kontextsensitiven Routingarchitektur zu finden und zu nutzen.

4. Zusammenfassung

Das im vorliegenden Kapitel beschriebene Architekturkonzept stellt die Basis für eine Unterstützung des kontextsensitive Routings dar. Der Aufbau und die Funktionsweise wurden ausführlich beschrieben. Das als Basis dienende AODV wurde beschrieben und den neuen Anforderungen entsprechend angepasst. Die Funktionen der drei Komponenten des Konzeptes – Client, kontextsensitiver Server und Kontextrouter – wurden detailliert dargestellt und sind so ausgelegt, dass auf mögliche Fehlerfälle entsprechend reagiert werden kann. Das Herzstück bildet der Kontextrouter, der einerseits seine Adresse im Netz verbreiten muss, die verfügbaren kontextsensitiven Server vorhält sowie die Auswahl eines passenden Servers für den Client übernimmt und andererseits schließlich die Kommunikation zwischen Client und Server unterstützt.

Die Anforderungen, die an das kontextsensitive Routing gestellt wurden, können mit dem vorliegenden Konzept realisiert werden. Es ist so angelegt, dass es in bestehende Netze eingepflegt werden kann, ohne deren Funktion zu gefährden. Die Kompatibilität wird weiterhin dadurch erhöht, dass ein Betrieb in heterogenen Netzen möglich ist. Das Konzept ist aktuell für IPv4 ausgelegt, kann aber für IPv6 übernommen werden. Die dazu notwendigen Modifikationen wurden bereits beschrieben.

Die Funktionen werden beim kontextsensitiven Routing auf der Vermittlungsschicht realisiert. Damit kann auf die Funktionen der darunter liegenden Schichten zurückgegriffen werden. Für höhere Schichten arbeitet das System transparent. Damit ist es auch für die kontextsensitiven Dienste transparent. Ausnahmen sind hierbei lediglich die für das Konzept nötigen Softwareerweiterungen beim Client und Server. Diese stellen neben der Routingerweiterung auch Schnittstellen zur Übergabe von Dienst- bzw. Anwendungsinformationen bereit. Des Weiteren ist auf eine möglichst große Flexibilität geachtet worden, damit gerade im Test und Realisierungsstadium schnell und einfach in die Funktionen einzelner Bestandteile (z. B. die Auswahllogik für die Server) eingegriffen werden kann.

Vorerst sind alle Komponenten der Architektur so ausgelegt, dass sie die Mindestfunktionalität für das kontextsensitive Routing erfüllen. Spätere Erweiterungen sind aber möglich. Damit ist das System noch ausbaufähig. So können beispielsweise Szenarien entwickelt werden, bei denen sich Kontextrouter netz- oder weltweit Informa-

tionen austauschen. Am Prinzip des kontextsensitiven Routings ändert sich dadurch aber nichts. Vollkommen unberücksichtigt sind bis dato Sicherheitsbetrachtungen. Da bei dem vorliegenden Konzept in der Regel auf bestehende Standards bzw. Spezifikationen zurückgegriffen wurde, kann an diesen Stellen von keinem erhöhten Risiko ausgegangen werden. Ob und inwieweit die kontextsensitive Routingarchitektur aber insgesamt neue Sicherheitslücken aufweist, muss noch untersucht werden. Es ist weiterhin zu erforschen, wie die Effizienz der Übertragung von Dienst und Kontexttypen durch Nutzung von Kodierungsverfahren erhöht werden kann. Teilweise wurde beim Konzept auf die Beschreibung konkreter Konfigurationswerte (z. B. Timeouts) verzichtet, da deren Größe flexibel in bestimmten Grenzen gewählt werden kann. Es sei an dieser Stelle auf die die in diesem Kapitel aufgeführten Standards verwiesen, die dazu eindeutige Vorgaben machen.

Literatur

[BRPD03] E. Belding-Royer, C. Perkins und S. Das. RFC 3561: Ad hoc On-Demand Distance Vector (AODV) Routing. Request for Comments 3561, Network Working Group, Juli 2003. Category: Experimental.

[Deer91] S. Deering. RFC 1256: ICMP Router Discovery Messages. Request for Comments 1256, Network Working Group, September 1991. STANDARDS-TRACK.

[KoPe02] Rajeev Koodli und Charles E. Perkins. Service Discovery in On-Demand Ad Hoc Networks. Internet draft, Seamoby Working Group, Oktober 2002.

[Pasc05] Guido Paschold. Entwurf und Simulation eines kontextsensitiven Routingprotokolls. Diplomarbeit, Technische Universität Ilmenau, Fachgebiet Kommunikationsnetze, Ilmenau, März 2005.

[PeRD00] Charles E. Perkins, Elizabeth M. Royer und Samir R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing for IP version 6. Internet draft, Mobile Ad Hoc Networking Working Group, November 2000.

[Perk02] C. Perkins. RFC 3344: IP Mobility Support for IPv4. Request for Comments 3344, Network Working Group, August 2002. Category: Standards Track.

[Renh06] Karsten Renhak. Router mit Kontextwissen. Studienarbeit, Technische Universität Ilmenau, Fachgebiet Kommunikationsnetze, Ilmenau, August 2006.

[RoPe99] Elizabeth M. Royer und Charles E. Perkins. Multicast Operation of the Ad-Hoc On-Demand Distance Vector Routing Protocol. In *Mobile Computing and Networking*, 1999, S. 207–218. citeseer.ist.psu.edu/article/royer99multicast.html.

[RoPe01] E. Royer und C. Perkins. Transmission range effects ao aodv multicast communication. Swedish Workshop on wireless Ad-Hoc Networks, März 2001. citeseer.ist.psu.edu/royer00transmission.html.

[Schi03] Jochen Schiller. *Mobilkommunikation*. Pearson Studium, München. 2. Auflage, 2003.

[Schm05] Enrico Schmidt. Adressierung für kontextsensitives Routing. Diplomarbeit, Technische Universität Ilmenau, Fachgebiet Kommunikationsnetze, Ilmenau, März 2005.

[Seni99] D. Senie. RFC 2644: Changing the Default for Directed Broadcast in Routers. Request for Comments 2644, Network Working Group, August 1999. Category: Best Current Practice.

[SrHo01] P. Srisuresh und M. Holdrege. RFC 3027: Protocol Complications with the IP Network Address Translator. Request for Comments 3027, Network Working Group, Januar 2001. Informational.